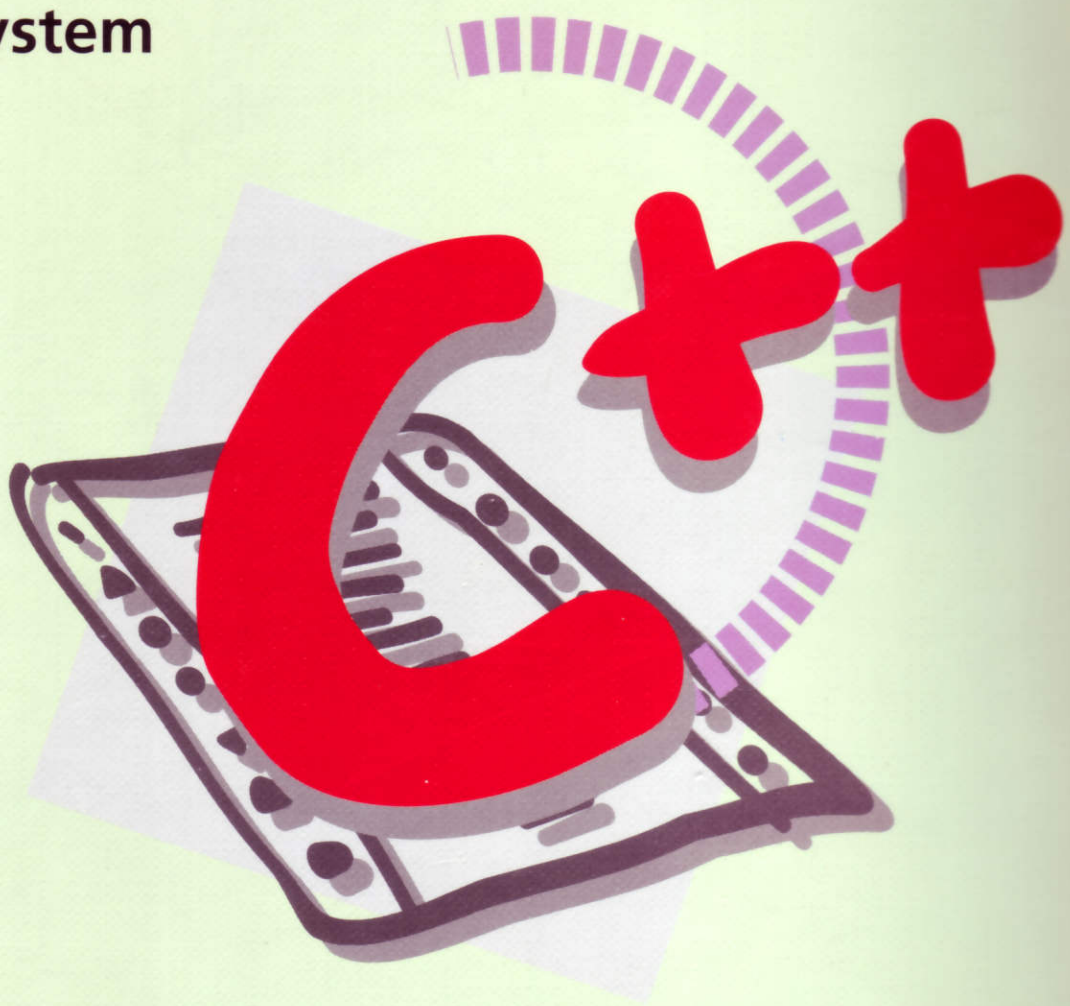
Maxon C++ 3

Integriertes Compilersystem



AMIGA®

MAXON computer

AMIGA

MaxonC++

Klassenbibliothek

EASY-OBJECTS

MAXON computer

2.3 Ein paar wichtige Definitionen in <new.h> ... 602
2.4 Ausnahmebehandlung: <exception.h> ... 605
2.5 Eine String-Library: <tools/str.h> ... 606
2.5.1 Die Klasse „String" ... 606
2.5.2 Member-Funktionen ... 607
2.5.3 Der Operator „+" auf Strings ... 609
2.5.4 Vergleiche ... 610
2.5.5 Ein- und Ausgabe ... 611
2.6 Interna von MaxonC++: <streamdefs.h> ... 612
2.6 Funktionen für originelle Linker-Opt.: <linkerfunc.h> ... 615

Easy-Objects ... 617
Einführung ... 619
Die Ausnahmebehandlung ... 623
Container und Cursor ... 623
Das Handlerkonzept ... 625
Die Fensterklasse WindowC ... 627
Die Fensterklasse GTWindowC ... 630
Gadgets und Gadgetlisten ... 630
Der Layouter ... 632
Die Regeln zur Bescheibung eines Layouts ... 633
Der Layoutvorgang ... 639
Eine Knopfzeile ... 639
Eine Gadgetspalte ... 641
Die Grenzen des Layouters ... 643
Zur Gestaltung der Oberfläche ... 643
Die Fensterklasse LayouterWindowC ... 644
Die Fensterklasse StandardWindowC ... 645
Datei- und Verzeichnissperren ... 646
Die Dateiklassen ... 647

Kommandozeilenargumente 648
Beschreibung des HotHelp Projekts zu EASY-OBJECTS 648
Einführung 648
Hintergrund 648
Technische Kapitel 649
Benutzung des HotHelp Projekts EASY-OBJECTS 651

Source Level Debugger 655

1. Einführung 657
Was können Sie mit MaxonSDB tun? 657
Über dieses Handbuch 657

2. Die Benutzung von MaxonSDB 659
Die Vorbereitung Ihres Programms 659
Programmstart 659
Der Workbenchstart 659
Der Shellstart 660
Der Start aus der Entwicklungsumgebung 661

3. Die Fenster des Debuggers 662
Drag&Drop 662
Das Projektfenster 662
Das Quelltextfenster 663
Breakpoints 664
Das Modulfenster 664
Das Fenster der globalen Variablen 665
Den Knopf Inspiziere. 665
Das Fenster der Funktionen 666
Das Fenster der lokalen Variablen 667
Das Fenster der aktuellen Variablen 667
Das Fenster zur Überwachung 667
Die Konfiguration der Oberfläche 668

Das Menü Projekt 669
Das Menü Schritt 669
Das Menü Fenster 670
Das Menü Einstellungen 670

Anhang 671
Der Enforcer 671
Der SegTracker 671
inline-Funktionen 671
Technische Einzelheiten: 672
Forbid() und Disable() 672
Die Prozessoren 68020,68030 und 68040 673
Sicherheitsabfragen 674
Fehlermeldungen 675

HotHelp 3 677

1. Über diese Anleitung 679

2. Jederzeit hilfsbereit 680

3. Eigenschaften von HotHelp 681

4. Anforderungen 683

5. Installation 683
5.1. Installation auf Festplatte 683
5.2. Installation auf Disketten 684
5.3. Der Installationsvorgang 685
5.4. Installation von Hand 686
5.5. Installation mit englischer Benutzerführung 687

6. Die Bedienung von HotHelp 688
6.1. Die Titelzeile des HotHelp-Fensters 689
6.2. Rollsymbole 690
6.3. Querverweise 690
6.4. Das Vortext-Symbol 691
6.5. Blättern im Projekt 691
6.6. Schlüssel, Projekte und Muster 691
6.7. Abrufen der Projektübersicht 692
6.8. Die Kapitelübersicht 693
6.9. Der Export von Hilfstexten 693
6.10. Verschiedene Ansichten über HotHelp 694
6.11. Die integrierte Suchfunktion 695
6.12. Verwendung von Lesezeichen 696
6.13. Hilfe über Hilfe 696
6.14. Tastaturbedienung 697
6.15. Möglichkeiten, HotHelp aufzurufen 697

7. Onlineprojekte 699
7.1. Benutzung der Onlineprojekte 699
7.2. Anwendungsgebiete für Onlineprojekte 700
7.3. Onlineprojekte und andere Editoren 700

8. ARexx 701

9. Änderungen seit Version 2 702

10. Die HotHelpTools 704
10.1. Die Hilfsfunktion der HotHelpTools 704
10.2. Start der HotHelpTools 705
10.3. HotHelpPref - HotHelps Voreinstellungen 706
10.3.1. Fenster 1: Farben 707

10.3.2. Fenster 2: Zeichensatz 708
10.3.3. Fenster 3: Ausgabeformate 709
10.3.4. Fenster 4: Steuerung 710
10.3.5. Fenster 5: Hotkeys 711
10.3.6. Fenster 6: Online-Optionen 711
10.3.7. Fenster 7: HKH-Optionen 712
10.4. HotHelpPro - Der Projektmanager 714
10.4.1. Standardprojekte 714
10.4.2. Onlineprojekte 715
10.4.3. Projekttest 716
10.4.4. AmigaGuide-Projekte 716
10.5. HotHelpMarker - der Projekte-Editor 718
10.5.1. Grundlagen der Projekterstellung 718
10.5.2. Hintergrund 719
10.5.3. Eingabe der Texte 719
Die Quelldateien 720
Die Texte 720
10.5.4. Markieren der Texte 720
Der HotHelp-Textmarker 720
Aufteilung des Textes 721
Einteilung in Einträge 721
Die Schlüssel 722
Querverweise 722
Grafiken 723
Schriftarten 723
Die Übersetzung 723
10.5.5. Resümee 724
10.5.6. Weitere Gestaltungsmöglichkeiten 724
Titel 725
Kapitel, Vorgänger und Nachfolger 725
Projekt- und ARexx-Querverweise 725
Schlüsselgruppen 725
10.6. EasyHotHelp 727
10.7. HotHelpComp - der HotHelp-Übersetzer 728
11. Letzte Worte 729

Einführung

Wie arbeitet man mit der Klassenbibliothek? Der Grundaufbau und das Hauptprogramm jeder Applikation, die eine graphische Benutzeroberfläche besitzen soll, sieht immer ziemlich ähnlich aus.

Als Beispiel wird hier das unvermeidbare HelloWorld-Programm beschrieben, allerdings erfolgt die Ausgabe in einem größenveränderbaren Fenster.

Diese drei Include-Anweisungen importieren einen Großteil der Klassenbibliothek...

```
#include <classes/layouter/windows.h>
#include <classes/layouter/boopsigadgets.h>
#include <classes/exec/libraries.h>
```

Beginnen wir mit der Deklaration eines Fensters. Dieses wird von der Klasse `StandardWindowC` abgeleitet, eine Fensterklasse, deren Fenster größenveränderbar, verschiebbar und schließbar sind. Außerdem wird automatisch der Zeichensatz des Bildschirms für die Gadgets verwendet.

```
class HelloWorldWindowC : public StandardWindowC {
public:
   HelloWorldWindowC(GTIDCMPortC &, ScreenC &);
   ~HelloWorldWindowC();
private:
```

Wir verwenden ein Textausgabefeld, eine Geometrie dafür und einen Handler für das Schließsymbol des Fensters.

```
   LBTextC hello;
   GeometryC helloGeo;
   WindowCloseHandlerC wch;
};
```

Bei solchen neuen Fenstern ist der Konstruktor die wichtigste Methode. Darin wird das Layout aufgebaut und die Handler des Fensters aktiviert.

```
HelloWorldWindowC::HelloWorldWindowC(GTIDCMPortC &p, ScreenC &s)
```

Das Standardfenster ist schon so, wie wir es wollen. Hier könnte man aber z.B. noch eine Größe angeben, oder festlegen, in welchem Rand das Sizegadget sitzt oder einen Titel des Fensters oder ...

```
   : StandardWindowC(p,s,NULL,
     TAG_END),
```

Das Textausgabefeld wird ohne Ereignisbehandlung initialisiert (es kann auch in diesem Beispiel keine Ereignisse sinnvoll behandeln, denkbar wäre allerdings eine Helpfunktion o.ä).

```
     hello(NULL,*this,LAYOUT_AUTOSIZE,LAYOUT_AUTOSIZE,
     LBA_Text,"Hello World!",
     LBA_Adjust,LB_AdjustCenter,
     TAG_END),
```

Die Geometrie initialisieren wir derart, daß das Gadget oben, unten, rechts und links an den Rändern der Gruppe hängt, d.h. das Gadget wird den gesamten inneren Bereich des Fensters füllen.

```
helloGeo(hello,
 LAYOUT_GROUP,NULL,2,LAYOUT_GROUP,NULL,-2,
 LAYOUT_GROUP,NULL,2,LAYOUT_GROUP,NULL,-2),
```

Der Handler des Schließsymbols hängt sich durch die Initialisierung (wobei ja die Referenz auf das Fenster angegeben wird) selbst in die Handlerkette des `GTIDCM` Ports ein und wird dadurch aktiv. Dieser Schließhandler beendet einfach das Programm.

```
    wch( *this)
{
```

Das Gadget wird in die Liste der Gadgets eingehängt und die Geometrie in die Geometriegruppe, die den inneren Bereich des Fensters umfasst. Da wären theoretisch auch noch die 4 Ränder des Fensters möglich. Diese Bereiche sollten allerdings Spezialfällen vorbehalten sein (z.B. Scrollern im unteren und rechten Rand des Fensters).

```
  gadgets.add(hello);
  innerGeo.add(helloGeo);
}
```

Wichtig

Der Konstruktor schließt das Fenster, damit die Gadgets nicht vor dem Schließen des Fensters freigegeben werden. Dies könnte zu Problemen führen, wenn in dieser Zeit noch ein letztes Refresh-Event kommt.

Unbedingt merken!!

```
HelloWorldWindowC::~HelloWorldWindowC()
{
  close();
}
```

Sämtliche Libraries, die für den Betrieb der Klassenbibliothek mindestens benötigt werden, werden geöffnet.

```
LibraryBaseErrC GadToolsBase("gadtools.library",37);
LibraryBaseErrC UtilityBase("utility.library",37);
LibraryBaseErrC CxBase("commodities.library",37);
LibraryBaseErrC LayersBase("layers.library",37);
LibraryBaseErrC WorkbenchBase("workbench.library",37);
```

Das Hauptprogramm sieht eigentlich so sehr typisch aus:

```
int main()
{
```

Zuerst wird überprüft, ob alle Libraries geöffnet werden konnten. Eine Fehlermeldung ist durch die Verwendung der Klasse `LibraryBaseErrC` allerdings schon ausgegeben, das Programm braucht nur abgebrochen zu werden.

```
  if (!LibraryBaseC::areAllOpen())
    return 20;
```

Das folgende Objekt ist das Kernstück des Programms, es empfängt und verarbeitet alle Signale (und damit auch Nachrichten, Semaphoren etc).

```
  SignalsC sc;
```

Jedes Fenster will seinen Bildschirm genau kennen. Der Einfachheit halber wird hier der Standardbildschirm gesperrt, ein "echtes" Programm sollte natürlich etwas flexibler sein.

```
  PublicScreenC screen();
  if (!screen.lock(NULL))
    return 100;
```

Alle `IDCMP` Nachrichten aller Fenster können durch einen einzigen Nachrichtenport empfangen werden. Dieser wird in der Variante für Gadtools-Fenster verwendet da die Fensterklasse `StandardWindowC` davon indirekt abgeleitet ist.

```
  GTIDCMPortC port;
  sc.add(port);
```

Das Fenster wird initialisiert.

```
  HelloWorldWindowC window(port,screen);
```

Das Programm kann durch den folgenden Handler auch noch mit <Ctrl>+<C> abgebrochen werden.

```
  CtrlCHandlerC ctrlchandler;
  sc.add(ctrlchandler);
```

Nun wird das Fenster geöffnet. Wenn das schief geht, wird das Programm durch eine Ausnahmebehandlung einfach abgebrochen. In einem "echten" Programm sollte dieser Fall abgefangen sein.

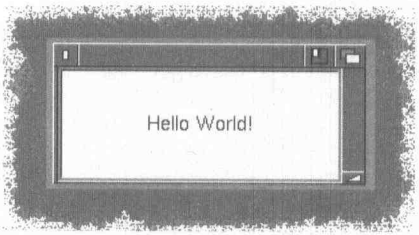
```
  window.open();
```

Nun wird die Signalbehandlung aktiviert. Das Programm befindet sich bis zum Ende in dieser Schleife und behandelt ankommende Signale.

```
  sc.loop();
```

Ende gut...

```
    return 0;
}
```



Wenn sie dieses Programm kompilieren und starten (es befindet sich in der Demo Schublade unter dem Namen HelloWorld.c), sehen sie ein kleines Fenster, in dem "Hello World!" ausgegeben wird. Sie können die Größe des Fensters verändern - der Text bleibt immer zentriert stehen. Beenden sie das Programm durch Drücken des Schließsymbols des Fensters.

Die Ausnahmebehandlung

Mit dem C++ Standard 3.0 wurden Ausnahmen (Exceptions) in die Sprache eingeführt. Diese Ausnahmen erleichtern die Fehlerbehandlung deutlich. EASY-OBJECTS verwendet die Ausnahmen allerdings nur für "echte" Fehler, d.h. bei Speichermangel oder Mangel anderer Ressourcen. Hingegen wird das Fehlschlagen des Öffnens einer Datei nicht als Ausnahme behandelt, schließlich ist es kein sehr seltener Fall, daß eine Datei geöffnet werden soll, die nicht existiert. In diesen und ähnlichen Fällen geben die entsprechenden Methoden Fehlercodes als Ergebnis zurück.

Die Ausnahmebehandlung der Bibliothek basiert auf einer Klassenhierarchie, die sämtliche möglichen Ausnahmen der Klassenbibliothek umfaßt. Einige Ausnahmen enthalten weitere Daten, um auf die Fehlerursache zurückzuschließen. Dabei werden grundsätzlich zwei Ausnahmearten unterschieden:

1. Ausnahmen, die durch eine falsche Verwendung der Klassen zurückzuführen sind. Dazu zählen z.B. fehlerhafte Layouts. Grundsätzlich sind diese Ausnahmen auf Programmfehler zurückzuführen und als Bug im Programm zu werten.

2. Ausnahmen, die durch fehlschlagende Ressourcenallozierung ausgelöst werden. Die meisten Versuche Speicher zu allozieren, z.B. für Systemstrukturen, führen zu Ausnahmen dieser Art, insbesondere werfen alle fehlschlagenden Konstruktoren Ausnahmen.

Die Klassenhierarchie enthält drei Schichten: alle Ausnahmen sind von der Klasse **`Exception`** abgeleitet. Diese abgeleiteten Klassen definieren funktionelle Teilbereiche, in die die konkrete Ausnahme fällt, z.B. finden sie alle Ausnahmen, die durch eine fehlschlagende Ressourceallozierung ausgeworfen werden, von der Teilbereichsklasse **`ResourceX`** abgeleitet. Von diesen Teilbereichsklassen sind nun direkt die konkreten Ausnahmen abgeleitet, diese enthalten zum Teil weitere Daten, um die konkrete Fehlersituation zu rekonstruieren.

Container und Cursor

Die Klassenbibliothek enthält verschiedene Klassen, deren Objekte andere Objekte sammeln können.

Die Exec Library stellt doppelt-verkettete Listen zur Verfügung, diese Listen stehen in zwei Varianten zur Verfügung: die Klasse **`ListC`** mit den Einträgen **`NodeC`** basiert auf der Struktur **`MinList`**, die Klassen **`EListC`** mit den Einträgen **`ENodeC`** basieren auf der Struktur **`List`**.

Weitere wichtige Container sind die dynamischen Arrays. Diese stehen sowohl in einer generischen Form, wie auch als Templates zur Verfügung. Von der Klasse **`gen_array`** bzw. **`array`** erben zwei Klassen, die auf diesen Arrays einige typische Listenoperationen ermöglichen, die Klassen **`gen_arraylist`** und **`arraylist`**.

Desweiteren gibt es einige Containerklassen für Spezialanwendungen, z.B. für Workbenchargumente und Tooltypes, die Liste öffentlicher virtueller Bildschirme und Taglisten.

EASY-OBJECTS verwendet ein einheitlichens Prinzip um die Objekte der Containerklassen zu verarbeiten. Zu jeder Containerklasse gehört eine passende Cursorklasse. Jeder Cursor stellt neben dem Konstruktor, dem sie mitteilen, welches Containerobjekt bearbeitet werden soll, zumindest 3 Methoden zur Verfügung: Zuerst einmal eine Methode `isDone()` zum Test, ob der Cursor schon am Ende des Containers steht, d.h. ob sämtliche Elemente bearbeitet wurden. Als nächstes eine Methode `next()`, durch die der Cursor auf das nächste Element des Containers gesetzt wird, und schließlich noch eine Methode `item()` mit dem auf das aktuelle Element, auf das der Cursor zeigt, zugegriffen wird.

Die meisten Cursor kennen noch eine Methode `first()` um den Cursor wieder auf den Anfang des Containers zurückzusetzen und die Listencursor (`ListCursorC` und `EListCursorC`) kennen noch die Methoden `last()` und `prev()`, um die Liste auch von hinten nach vorne zu durchsuchen.

Beispiel:

```
arraylist<int> zahlen;
arraycursor<int> zaehler;

zahlen.addTail() = 6;
zahlen.addTail() = 8;
zahlen.addTail() = 42;

while (!zaehler.isDone())
{
   cout << zaehler.item() << "\n";
   zaehler.next();
};
```

Dieses Beispiel legt die drei Zahlen 8, 6 und 42 auf eine Liste, die allerdings über ein dynamisches Array implementiert ist. Dann werden über die Verwendung des Cursors die drei Zahlen wieder ausgegeben.

Der Vorteil der Verwendung der Cursor liegt klar auf der Hand: Egal, ob Sie doppelt-verkettete Listen, dynamische Arrays, Workbenchargumente, Tooltypes oder Taglisten verarbeiten, der Zugriff erfolgt immer identisch und Sie können solche Schleifen sehr viel schneller und weniger fehleranfällig programmieren.

Das Handlerkonzept

Auftretende Ereignisse werden durch Signale angezeigt. Diese Ereignisse können sehr unterschiedlicher Natur sein, es kann z.B. das Signal <CTRL>+<C> direkt gesetzt werden, oder eine Nachricht an einem Nachrichtenport eintrudeln.

Um diese Ereignisse zu verarbeiten, werden Signalhandler eingesetzt. Die Klasse **SignalsC** verwaltet eine Liste von Signalhandlern (ein neuer Handler wird mit der Methode **SignalsC::add** angehängt), und kann diese Signale verarbeiten, indem bei einem ankommenden Signal jeder Handler gefragt wird, ob er das Signal verarbeiten kann. Ist dem so, wird eine Behandlungsmethode des Handlers aufgerufen, die das Signal verarbeitet – z.B. die nächste Nachricht vom Port holt, diese irgendwie bearbeitet und an den Sender zurückschickt.

Das folgende Beispiel benutzt einen Handler für <Ctrl>+<C>. Dieser Handler reagiert auf das Setzen des entsprechenden Signals.

```
int main()
{
   SignalsC sc;

   CtrlCHandlerC ctrlc();
   sc.add();

   sc.loop()

   return 0;
}
```

Bleibt die Frage, warum das Programm beendet wird, wenn <Ctrl>+<C> gedrückt wird. Nun, die Methode, die das Signal verarbeitet, kann als Ergebnis zurückgeben, daß die Methode **SignalsC::loop** verlassen werden soll. Genau dies und nichts mehr macht die Klasse **CtrlCHandlerC** und damit wird das Programm auch beendet.

Für Nachrichtenports funktioniert dieses Handlerkonzept auch. Die meisten Nachrichten, die an einen Port geschickt werden, besitzen eine identische Struktur und lassen sich anhand eines Feldes in Ereignisklassen unterteilen. Ein gutes Beispiel sind die IDCMP Nachrichten; anhand des Feldes **Class** werden die Nachrichten in die verschiedenen Nachrichtenklassen unterteilt (z.B. **IDCMP_CLOSEWINDOW**, **IDCMP_RAWKEY** etc). Was liegt also näher, als wiederum Handler für jede Nachrichtenklasse einzuführen?

Die Klasse **PortC** beschreibt die Nachrichtenports, die Nachrichten empfangen können. Die Methoden **SignalHandlerC::forMe** und **SignalHandlerC::handle** sind in dieser Klasse geeignet überladen: Die Methode **PortC::forMe** liefert **TRUE**, sobald eine Nachricht am Nachrichtenport bereitliegt (beachtet also die Signalmaske nicht) und die Methode **PortC::handle** holt alle anliegenden Nachrichten vom Port und übergibt sie nacheinander der Methode **PortC::handleMsg**. Diese Methode muß die Nachricht verarbeiten und u.U. an den Sender zurückschicken.

Die Klasse **HandlerPortC** baut auf der Klasse **PortC** auf und erweitert diese um das Handlerkonzept für Nachrichten: Aufbauend auf der Klasse **MessageHandlerC** werden einem Nachrichtenport dieser Klasse Nachrichtenhandler übergeben (**HandlerPortC::add**).

Erhält ein Nachrichtenport der Klasse **HandlerPortC** eine Nachricht, so wird jeder Nachrichtenhandler gefragt, ob er diese Nachricht brauchen kann (**MessageHandlerC::forMe**). Falls dem so ist, wird die Methode **MessageHandlerC::handle** des Handlers aufgerufen, die diese Nachricht verarbeitet:

Das Verlassen der Methode **SignalsC::loop** wird hier im Unterschied zu den Signalhandlern nicht durch das Ergebnis der Behandlungsmethode der Nachricht ausgelöst, sondern durch eine eigene Methode **MessageHandlerC::exit**, die aufgerufen wird, wenn die Behandlungsmethode **MessageHandlerC::handle** das Ergebnis **TRUE** liefert.

Die Fensterklasse WindowC

Die Fensterklasse `WindowC` enthält die Fenster der Intuition Library. Diese Fenster können in einem offenen und geschlossenen Zustand existieren (im Gegensatz zu Intuition, wo man die Fensterstruktur erst durch das Öffnen des Fensters erhält).

```
#include <classes/intuition/window.h>
#include <classes/intuition/screen.h>
#include <classes/exec/libraries.h>

class TestWindowC : public WindowC {
public:
   TestWindowC(IDCMPortC &, STRPTR title);
   ~TestWindowC();
private:
   WindowCloseHandlerC wch;
};
```

Aussehen und Verhalten der Fenster wird mit den gleichen Tags festgelegt wie unter Intuition (also alle Tags die mit `WA_` beginnen). Eine Ausnahme stellt das Tag `WA_Gadgets` dar, die Gadgetverwaltung geschieht über Gadgetlisten, für deren Anbindung an das Fenster einige Methoden zur Verfügung stehen. Deshalb darf dieses Gadget nicht angegeben werden (bzw. wird unbarmherzig aus der Tagliste gefiltert).

```
TestWindowC::TestWindowC(IDCMPortC &p, STRPTR title)
   : WindowC(p,
      WA_DragBar,TRUE,
      WA_Width,300,
      WA_Height,150,
      WA_CloseGadget,TRUE,
      WA_DepthGadget,TRUE,
      WA_Title,title,
      WA_IDCMP,IDCMP_GADGETUP,
      TAG_END),
```

Die Tags können sowohl im Konstruktor, wie auch in der Methode `WindowC::open` angegeben werden

- Tags, die in der Open-Methode verwendet werden, haben nur für diesen einen Open-Befehl Wirkung.
- Tags, die im Konstruktor angegeben werden, haben für jeden Open-Befehl Wirkung.
- Einige Methoden verändern die Tagliste, die im Konstruktor angegeben wurde; damit können Methoden, die auch Sinn bei geschlossenem Fenster machen, auch dann verwendet werden. Dazu zählt das Setzen des Fenstertitels, des Bildschirmtitels und die Größenveränderung des Fensters.

Die IDCMP-Flags können sowohl über die Tags, als auch über die entsprechenden Methoden `WindowC::setIDCMP`, `WindowC::addIDCMP`, `WindowC::subIDCMP` gesetzt werden.

Zur Verarbeitung der Fenster-spezifischen IDCMP Nachrichten steht eine Klasse **WindowEventHandlerC** zur Verfügung. Objekte dieser Klasse verarbeiten Nachrichten einer bestimmten IDCMP Ereignisklasse, die zu einem bestimmten Fenster gehören. Dieses Fenster wird im Konstruktor des Handlers angegeben. Der Konstruktor macht jedoch noch mehr: Er addiert das entsprechende IDCMP- Flag zum Fenster und hängt sich selbst in die Liste der Handler des IDCMP Nachrichtenports des Fensters ein.

```
    wch( *this)
{
```

Für Handler, die nicht an ein bestimmtes Fenster gebunden sind, z.B. die Handler **GadgetUpHandlerC** und **GadgetDownHandlerC** zur Verarbeitung der Ereignisklasse **IDCMP_GADGETUP** und **IDCMP_GADGETDOWN** müssen die IDCMP-Flags extra im Fenster eingetragen werden.

```
  addIDCMP(IDCMP_GADGETUP|IDCMP_GADGETDOWN);
}

TestWindowC::~TestWindowC()
{
  close();
}
```

Fenster können "busy" gesetzt werden; dann wird der Mauszeiger für das Fenster auf die Uhr umgesetzt (unter V39 wird der Voreinsteller beachtet) und das Fenster nimmt keinerlei Eingaben mehr an. In diesem Beispiel wird dieser Status über die Fensterliste gesetzt (s.u.).

Die Fensterklasse **WindowC** stellt zwei Methoden zur Verfügung, um den Zustand des Fensters auch über die Lebensdauer eines Fensterobjekts zu speichern. Die Klasse **WindowSnapshotC** stellt Objekte zur Verfügung, die die Position und Größe sowie den Busy-Zustand des Fenstesr speichern können. Mit der Methode **snapshot** schreibt das Fenster seinen Zustand auf ein Objekt der Klasse **WindowSnapshotC**, mit der Methode **stamp** übernimmt ein Fenster diesen Zustand aus einem solchen Zustand. Die Daten des Snapshot-Objekts sind einfache Datentypen und können daher einfach in einer Voreinstellungsdatei gespeichert werden, um den Fensterzustand auch über den Programmlauf hinweg zu retten.

Hier folgt nun noch das Hauptprogramm, um die oben definierte Fensterklasse `TestWindowC` zu verwenden:

```
LibraryBaseErrC GadToolsBase("gadtools.library",37);
LibraryBaseErrC UtilityBase("utility.library",37);
LibraryBaseErrC CxBase("commodities.library",37);
LibraryBaseErrC LayersBase("layers.library",37);
LibraryBaseErrC WorkbenchBase("workbench.library",37);

int main()
{
  if (!LibraryBaseC::areAllOpen())
    return 20;

  // diese Klasse empfängt und verarbeitet alle Signale
  SignalsC sc;
```

IDCMP Nachrichten für alle Fenster können über einen einzigen Nachrichtenport empfangen werden. Die Klasse `IDCMPortC` kann die IDCMP Nachrichten empfangen und durch spezielle IDCMP Ereignishandler verarbeiten. Ein Beispiel ist die Klasse `WindowCloseHandlerC`, deren Objekte das Ereignis `IDCMP_CLOSEWINDOW` verarbeiten (und im Normalfall die Applikation beenden). Die Klasse `TestWindowC` verwendet diesen Handler.

```
  IDCMPortC port;
  sc.add(port);

  // einige Fenster
  TestWindowC window1(port, "Fenster 1: busy");
  TestWindowC window2(port, "Fenster2");
  TestWindowC window3(port, "Fenster 3: busy");
```

Objekte der Fensterklasse `WindowC` können in einer Liste verwaltet werden. Die Klasse `WindowListC` stellt dazu geeignete Objekte zur Verfügung. Fenster, die in einer solchen Liste eingetragen sind, können alle auf einmal geschlossen werden, oder alle, bis auf eine eventuelle Ausnahme, in den "busy" Zustand versetzt werden. Dadurch lassen sich auf einfache Weise modale Dialoge realisieren (im allgemeinen sollte man jedoch versuchen, modale Dialoge zu vermeiden).

```
  WindowListC windowliste;
  windowliste.add(window1);
  windowliste.add(window2);
  windowliste.add(window3);

  // alle Fenster öffnen
  window1.open(WA_Left,0,WA_Top,0,TAG_END);
  window2.open(WA_Left,20,WA_Top,20,TAG_END);
  window3.open(WA_Left,40,WA_Top,40,TAG_END);

  windowliste.setBusy(TRUE,&window2);

  sc.loop();
```

```
    windowliste.close();

    return 0;
}
```

Die Fensterklasse GTWindowC

Die Klasse **GTWindowC** unterscheidet sich äußerlich nicht von der Klasse **WindowC**, der Unterschied findet sich in der Eigenart der GadTools Library, die eine geringfügig andere Implementation des Fensteröffnens und -refreshs verlangt. Ebenso erwarten Fenster der Klasse **GTWindowC** einen IDCMP Nachrichtport der Klasse **GTIDCMPortC**, da auch der Nachrichtenempfang durch die GadTools-Library eine andere Implementation bedingt.

Gadgets und Gadgetlisten

Die Klassenbibliothek unterstützt die beiden gebräuchlichen Arten der Gadgets: BOOPSI Gadgets und GadTools Gadgets. Die Klasse **GadgetC** bildet dazu eine Basis für alle Gadgets und definiert damit ein technisches Modell für alle Gadgets.

Dieses technische Modell erlaubt eine generelle Behandlung der Position und Größe des Gadgets, Auswahl des Zeichensatzes für Textausgaben des Gadgets und einige Methoden zur Verwaltung in Gadgetlisten.

Ein abstraktes Gadget ist die Gadgetliste. Die Klasse **GadgetListC** erbt von der Klasse **GadgetC**, dadurch können Bäume von Gadgets gebildet werden. Die Gadgetliste nimmt weitere Gadgets der Klasse **GadgetC** auf und erlaubt, die Anwendung verschiedener Methoden auf alle Gadgets der Liste (z.B. den Zeichensatz zu ändern).

Neben diesem technischen Modell existiert noch das logische Modell der verschiedenen Gadgettypen. Jedes Gadget zeigt ein bestimmtes Verhalten, das unabhängig davon ist, ob es ein GadTools oder ein BOOPSI Gadget ist, oder wie das Verhalten konkret in Aussehen und Bedienung umgesetzt ist. Z.B. enthält die Klassenbibliothek ein BOOPSI-Cycle-Gadget, das die Auswahl über ein Popup-Menü erlaubt. Obwohl sich darin die Bedienung für den Benutzer sichtlich vom Cycle-Gadget der GadTools-Library unterscheidet, bleibt die Bedienung der Gadgets durch den Programmierer identisch. Also sind die beiden Gadgets (das Cycle-Gadget der GadTools-Library und das neue BOOPSI-Gadget) von dem gleichen logischen Modell abgeleitet: der Klasse **GCycleC**.

Jedes Gadget der GadTools-Library und viele der neuen BOOPSI-Gadgets erben also nicht nur von der Klasse **GadgetC**, sondern auch von einer der logischen Modellklassen (die alle von **GModelC** erben).

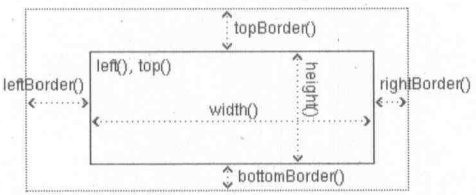
Neben diesen beiden Klassen, die die Basis eines jeden Gadgets bilden, verwenden die Gadgets ein Objekt der Klasse **GadgetEventC**, das die Ereignisbehandlung des Gadgets festlegt. Ein Gadget kann verschiedene Ereignisse auslösen, z.B. ein **IDCMP_GADGETUP** Ereignis, oder - über einen speziellen Handler - ein **IDCMP_RAWKEY** Ereignis. Diese Ereignisse werden von passenden Handlern

aufgefangen. Die Handler rufen dann die zu dem Ereignis passende Methode der Klasse **GadgetEventC** auf, z.B. die Methode **GadgetEventC::up** im Falle eines **IDCMP_GADGETUP** Ereignisses.

Die Fensterklasse **WindowC** stellt einige Methoden zur Verfügung, mit denen eine Gadgetliste und ein Fenster assoziiert werden. Gadgets dürfen in ein Fenster der Klasse **WindowC** nur über eine solche Fensterliste eingetragen werden, das Tag **WA_Gadgets** ist nicht erlaubt (und wird durch die Methode **WindowC::open** ausgefiltert).

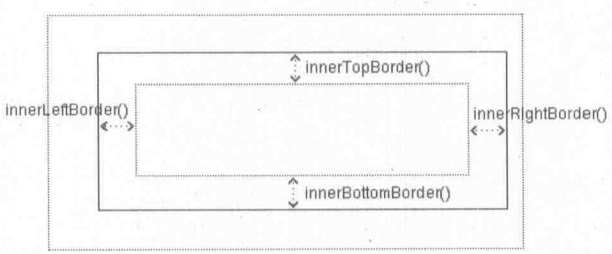
Der Layouter

Der Layouter positioniert Rechtecke auf einer rechteckigen Grundfläche. Dabei wird sowohl die Position wie die Größe der Rechtecke berechnet. Jedes Rechteck legt darüberhinaus einen Rahmen fest, der das Rechteck umgibt. Die Breite dieses Rahmens, der natürlich an allen vier Seiten des Rechtecks unterschiedlich breit sein kann, wird durch das Rechteck bestimmt, nicht durch den Layouter! Die folgende Abbildung zeigt ein Rechteck mit Rahmen und einige Gadgets mit unterschiedlichen Rahmen, die durch die Position und Größe der Beschriftungen festgelegt sind.



Das Modell eines solchen Rechtecks für den Layouter bildet die Klasse **`LayoutC`**. Sie enthält Methoden zur Positionierung und Größenveränderung des Rechtecks und der Ermittlung der Rahmengröße.

Neben diesem Grundmodell kennt der Layouter noch eine erweitere Form: das Gruppenrechteck. Diese Form steht für Rechtecke, in denen selbst wieder weitere Rechtecke liegen können. Dieses Rechteck kennt neben der Position, Größe und dem äußeren Rahmen auch noch einen inneren Rahmen. Dieser Rahmen bestimmt den Abstand der innenliegenden Rechtecke zum Rand. Die folgende Abbildung zeigt ein Gruppenrechteck mit äußeren und inneren Rahmen.



Das Modell eines Gruppenrechtecks für den Layouter bildet die Klasse **`GroupLayoutC`**, die von der Klasse **`LayoutC`** erbt. Sie enthält zusätzliche Methoden zur Ermittlung der inneren Rahmenbreite und zur Bestimmung einer Minimalgröße des Rechtecks.

Diese beiden Modelle werden durch den Layouter verwendet, um die Elemente (Gadgets) zu positionieren und in der Größe zu verändern. Um zum Beispiel einen Button zu erhalten, der durch den Layouter verwendet werden kann, muß man die Klasse des Buttons (z.B. **`BIButtonC`**) zusammen mit der Klasse **`LayoutC`** zu einer Klasse „layoutbarer" Buttons vererben (z.B. **`LBIButtonC`**). Die

abstrakten Methoden der Klasse `LayoutC` sind dann passend mit den Methoden zur Positionierung und Größenveränderung der Buttonklasse zu überladen.

Die „layoutbaren" Gadgets akzeptieren alle einen besonderen Wert für die Breite und Höhe des Gadgets: `LAYOUT_AUTOSIZE`. Wenn die Größe der Gadgets festgelegt wird (durch die Methoden `LayoutC::layoutWidth` und `LayoutC::layoutHeight`), erkennen die Gadgets, wenn sie mit dieser speziellen Größenangabe konstruiert wurden. Daraufhin berechnen sie die nötige Größe für sich selbst automatisch.

Die Buttonklasse enthält nun Objekte, die durch den Layouter verarbeitet werden können. Allerdings werden damit noch keine Regeln festgelegt, die das Layout beschreiben, d.h. die Position und Größe des Buttons in Abhängigkeit der anderen Elemente auch tatsächlich festlegen.

Diese Regeln werden durch Objekte der Klassen `GeometryC` und `GeometryGroupC` festgelegt. Zu jedem Objekt der Klasse `LayoutC` gehört ein Objekt der Klasse `GeometryC`, und zu jedem Objekt der Klasse `GroupLayoutC` ein Objekt der Klasse `GeometryGroupC`.

Durch die Trennung der Elemente, die durch den Layouter verarbeitet werden von der Geometrie eines Elements können die Regeln des Layouts von den in diesem Layout verwendeten Elementen unabhängig entwickelt werden. Zum Beispiel sind dadurch Bausteine für das Layout denkbar, die immer wieder eingesetzt werden können.

Wie wird nun eine Geometrie eines Rechtecks festgelegt? Im Prinzip ganz einfach: Ein Rechteck hat 4 Koordinaten (links, rechts, oben, unten) und entsprechend wird für jede Koordinate eine Regel angegeben. Diese Regel bestimmt in irgendeiner Art den Wert füt diese Koordinate, z.B. kann dadurch die Koordinate sich auf die Koordinate eines anderen Elements beziehen, oder auf den Rand des Gruppenrechtecks, in dem das Rechteck liegt.

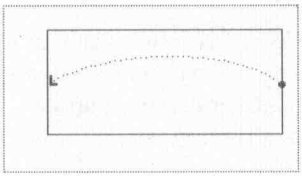
Die Regeln zur Bescheibung eines Layouts

Die Regeln werden im Konstruktor der Klassen `GeometryC` und `GeometryGroupC` angegeben. Jede Regel für jede der 4 Koordinaten besteht aus 3 Teilen:

- Einer der 14 Bezeichner für die Regelart.
- Ein Pointer auf ein Objekt der Klasse `GeometryBaseC`, üblicherweise ein Objekt der Klasse `GeometryC` oder `GeometryGroupC` oder `NULL`.
- Eine Zahl, die entweder einen Pixeloffset oder eine Prozentzahl angibt; da der Pixeloffset immer addiert wird, ist er in Regeln für die rechte oder untere Koordinate normalerweise negativ.

Die 14 Bezeichner bedeuten im einzelnen:

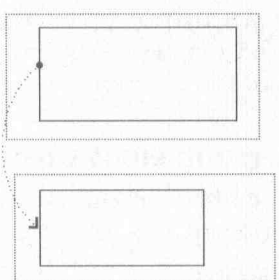
LAYOUT_SIZE



Jedes Gadget kennt eine eigene Größe. Diese kann entweder bei der Initialisierung des Gadgets angegeben, oder durch das Gadget selbst berechnet werden (typischerweise durch Angabe des speziellen Werts `LAYOUT_AUTOSIZE` für die Breite oder Höhe des Gadgets).

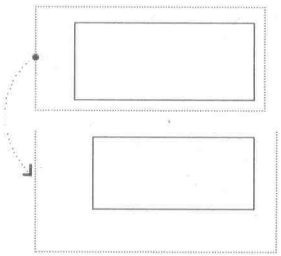
Diese Regel bedeutet, daß die entsprechende Koordinate sich im Abstand der Größe des Gadgets auf die Position der gegenüberliegenden Koordinate bezieht. Wenn Sie also für die linke Koordinate diese Regel angeben, so wird damit das Gadget so breit wie das Gadget eben ist, wobei die rechte Koordinate in einem gültigen Layout die Position festlegen muß. Deshalb können nicht beide Koordinaten nach dieser Regel berechnet werden.
Der Pointer muß `NULL` und der Offset `0` sein.

LAYOUT_SAME



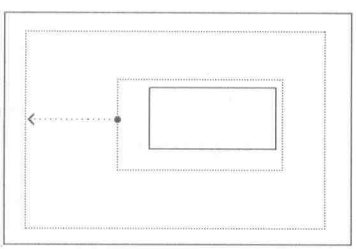
Diese Regel legt die Koordinate auf den gleichen Wert der gleichen Koordinate eines anderen Rechtecks. Zu dieser Regel müssen Sie die Geometrie des anderen Rechtecks angeben und eine Distanz, die zusätzlich zu der Koordinate addiert wird.

LAYOUT_SAMEBORDER



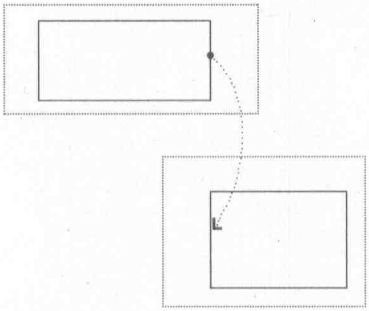
Diese Regel ist im Prinzip der oberen gleich, beachtet jedoch zusätzlich den Rand der beiden Rechtecke. Durch diese Regel liegen die Ränder beider Rechtecke auf der gleichen Position, nicht die Koordinate des Rechtecks selbst. Wiederum kann eine zusätzliche Distanz angegeben werden.

LAYOUT_GROUP



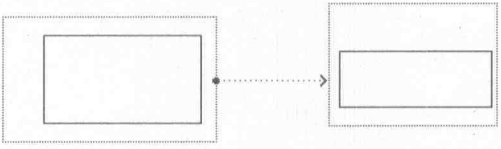
Diese Regel bezieht den Rand des Rechtecks auf den inneren Rand des umgebenden Rechtecks. Eine sehr häufig gebrauchte Regel. Es kann eine Distanz zum Rand angegeben werden.

LAYOUT_OPP



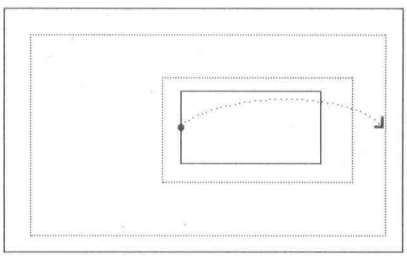
Diese Regel entspricht der Regel **LAYOUT_SAME**, bezieht sich allerdings auf die gegenüberliegende Koordinate des anderen Rechtecks. Es kann eine zusätzliche Distanz zwischen den Objekten angegeben werden.

LAYOUT_OPPBORDER



Diese Regel entspricht der Regel **LAYOUT_SAMEBORDER**, bezieht sich allerdings auf die gegenüberliegende Koordinate des anderen Rechtecks. Damit lassen sich sehr einfach Rechtecke in eine Zeile nebeneinander oder in eine Spalte untereinander legen. Es kann eine zusätzliche Distanz zwischen den Objekten angegeben werden.

LAYOUT_OPPGROUP



Diese Regel entspricht der Regel `LAYOUT_GROUP`, bezieht sich aber auf die gegenüberliegende Seite der umgebenden Gruppe. Da nur die Distanz dann noch einen Abstand zum Rand schaffen kann, ist das eine eher selten gebrauchte Regel.

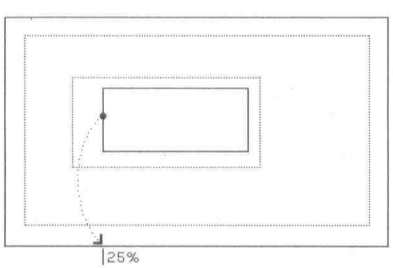
LAYOUT_MAXSIZE

Diese Regel ermittelt die Größe aller Rechtecke der Gruppe, und setzt die Koordinate in diesem Abstand zur Position der gegenüberliegenden Koordinate. Der Offset kann einen Minimalwert für die Größe angeben.

LAYOUT_SAMESIZE

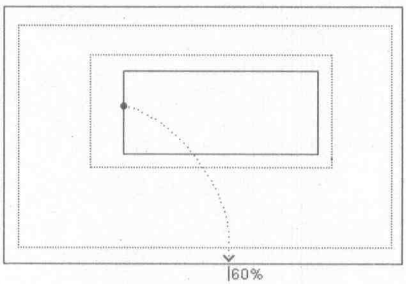
Diese Regel ermittelt die Größe des anderen Rechtecks, nachdem dieses bearbeitet wurde, und setzt die Koordinate in diesem Abstand zur Position der gegenüberliegenden Koordinate. Der Offset muß 0 sein.

LAYOUT_PROCENT



Diese Regel ermittelt die Breite des umgebenden Rechtecks und setzt die Koordinate auf einen Prozentteil dieser Breite. Der Prozentteil muß statt einer Distanz angegeben werden.

LAYOUT_PROCENTCENTER



Diese Regel ermittelt die Breite des umgebenden Rechtecks und des eigenen Rechtecks. Dann setzt sie die Koordinate so, daß die Mitte des Rechtecks auf dem angegebenen Prozentteil der Breite des umgebenden Rechtecks zu liegen kommt.

LAYOUT_MAXSAMEBORDER

Diese Regel ermittelt die maximale Breite des entsprechenden Randes aller Rechtecke und setzt die Koordinate in diesem Abstand zur Position der gegenüberliegenden Koordinate.

LAYOUT_MAXOPPBORDER

Diese Regel entspricht der oberen Regel, berechnet allerdings die maximale Größe des gegenüberliegenden Randes aller Rechtecke.

LAYOUT_HOOK

Obwohl das nun ein ganzer Haufen Regeln sind, könnte es ja noch nicht ausreichen. Tatsächlich gibt es z.B. ein Gadget, mit dem das Layout durch den Benutzer manuell verändert werden kann. Um solche besonderen Regeln zu implementieren, dient diese Regel. Die Verwendung dieser Regel ist allerdings noch privat (d.h. nicht dokumentiert).

Durch Kombination dieser Regeln ist so gut wie jedes "schöne" Layout erstellbar. Alle Regeln beachten die Einteilung der Geometrien in die Gruppen. Wenn also die beiden Prozentregeln den Prozentanteil an der Breite oder Höhe ermitteln, so wird als Berechnungsgrundlage die Breite oder Höhe der Gruppe verwendet, in der die Geometrie, die die Prozentregel benutzt, liegt. Gleiches gilt auch für die Regeln, die die maximale Größe oder Rahmenbreite ermitteln.

Dennoch können mutige Layouter über Gruppengrenzen hinweg Elemente ansprechen. Es ist allerdings nicht gesagt, ob das immer funktioniert. Im wesentlichen kommt das auf die exakte Abhängig-

keit der Elemente untereinander an. Im Zweifel einfach ausprobieren. Übrigens: Jedes Layout, daß mit einem Zeichensatz korrekt aussieht, sollte es auch mit jedem anderen tun - insofern Sie keine Pixelangaben für die Größen der Elemente verwendet haben.

Der Layoutvorgang

Der Layouter enthält eine Klasse `RootGeometryC`. Sie bildet die Wurzel des Geometrienbaums.

Diese Klasse enthält eine wichtige Methode: `RootGeometryC::layout`. Damit wird das Layout in ein Rechteck, dessen linke, obere Ecke und Größe angegeben wird, eingefügt.

Diese Methode akzeptiert den speziellen Wert `LAYOUT_AUTOSIZE` für die Größe des Rechtecks. Dann wird die minimal benötigte Größe für das Layout berechnet. Beim ersten Layoutvorgang oder nach wesentlichen Änderungen des Layout (z.B. Änderungen am Geometrienbaum, oder Änderungen an den Gadgets, die die Ränder größe oder automatische Größe der Gadgets beeinflussen (Fontänderung!)) muß zuerst ein Layout mit dieser speziellen Größenangabe berechnet werden.

Damit tatsächlich die minimale Größe berechnet wird, sollten Sie für alle Gadgets, deren Breite oder Höhe durch den Layoutvorgang bestimmt wird, und nicht durch das Gadget selbst (also z.B. linke und rechte Koordinate haben die Regel `LAYOUT_GROUP`) für die entsprechende Größenangabe den Wert `LAYOUT_AUTOSIZE` verwenden. Tun Sie dies nicht, erkennt das Gadget bei wiederholten Layoutvorgängen nicht, daß es seine eigene Größe automatisch berechnen soll und nimmt deshalb die vom Layouter berechnete. Diese kann allerdings inzwischen (z.B. durch Veränderung der Größe des Fensters, in dem das Layout liegt) sehr groß geworden sein - der Layoutvorgang würde also nicht die minimale Größe, sondern eine sehr viel größere berechnen.

Eine Knopfzeile

Dieses Beispiel zeigt den Aufbau eines Bausteins für den Layouter: Häufig stehen am unteren Rand eines Requesters für Voreinstellungen drei Knöpfe: **"Speichern"**, **"Benutzen"**, **"Abbrechen"**. Dieser Baustein positioniert diese 3 Knöpfe in einer Zeile:

```
class ThreeButtonGroupC : public GeometryGroupC {
public:
```

Dem Konstruktor wird ein Objekt für den Rahmen, drei Knöpfe und die Regeln, wie dieser Baustein ins Layout eingepasst wird, übergeben.

```
    ThreeButtonGroupC(
      GroupLayoutC &frame,
      LGButtonC &left,
      LGButtonC &middle, LGButtonC &right,
      ULONG topRelation, GeometryC *topObject, WORD topOffset,
      ULONG bottomRelation, GeometryC *bottomObject, WORD bottomOffset,
      ULONG leftRelation, GeometryC *leftObject, WORD leftOffset,
      ULONG rightRelation, GeometryC *rightObject, WORD rightOffset);
private:
  GeometryC leftGeo;
```

```
    GeometryC middleGeo;
    GeometryC rightGeo;
};
```

Der Konstruktor macht nun nichts mehr, als die Gruppe aufzubauen:

```
ThreeButtonGroupC::ThreeButtonGroupC(
      GroupLayoutC &frame,
      LGButtonC &left,
      LGButtonC &middle, LGButtonC &right,
      ULONG topRelation, GeometryC *topObject, WORD topOffset,
      ULONG bottomRelation, GeometryC *bottomObject, WORD bottomOffset,
      ULONG leftRelation, GeometryC *leftObject, WORD leftOffset,
      ULONG rightRelation, GeometryC *rightObject, WORD rightOffset)
    : GeometryGroupC(frame,
      topRelation, topObject, topOffset,
      bottomRelation, bottomObject, bottomOffset,
      leftRelation, leftObject, leftOffset,
      rightRelation, rightObject, rightOffset),
```

Der linke Knopf wird um die 20 Prozentmarke zentriert, erhält die Maximalgröße der 3 Knöpfe, aber mindestens 40 Pixel breit - dadurch wirken Knöpfe mit sehr kurzen Aufschriften ("Yes", "No", "OK") angenehmer.

```
      leftGeo(left,
      LAYOUT_SIZE,NULL,0,LAYOUT_GROUP,NULL,0,
      LAYOUT_PROCENTCENTER,NULL,20,LAYOUT_MAXSIZE,NULL,40),
```

Der mittlere Knopf wird um die 50 Prozentmarke zentriert. Beachten Sie die Angabe der Geometrie des linken Knopfes in der Prozentregel: Diese wird zur Größenberechnung des Platzbedarfs dieser Gruppe benötigt; die Regel schafft dann mindestens soviel Platz, daß die beiden Knöpfe sauber nebeneinander liegen.

```
      middleGeo(middle,
      LAYOUT_SIZE,NULL,0,LAYOUT_GROUP,NULL,0,
      LAYOUT_PROCENTCENTER,&leftGeo,50,LAYOUT_MAXSIZE,NULL,0),
```

Der rechte Knopf wird symmetrisch zum linken um die 80 Prozentmarke zentriert. Auch hier wird wiederum der linke Nachbar, also der mittlere Knopf in der Prozentregel zur Platzberechnung angegeben.

```
      rightGeo(right,
      LAYOUT_SIZE,NULL,0,LAYOUT_GROUP,NULL,0,
      LAYOUT_PROCENTCENTER,&middleGeo,80,LAYOUT_MAXSIZE,NULL,0)
{
    GeometryGroupC::add(leftGeo);
    GeometryGroupC::add(middleGeo);
    GeometryGroupC::add(rightGeo);
}
```

Dieser Baustein ist nun universell für solche 3-Knopf-Zeilen verwendbar. Manchmal ist er vielleicht zu universell. Man könnte z.B. eine neue Gruppe ableiten, die gleichzeitig von einer Gadgetliste erbt, und die Gadgets für die Knöpfe und den Rahmen der Gruppe darin festlegen und zu dieser Liste addieren. Wie allgemein oder speziell Sie ihre Gruppen aufbauen, kommt auf den Kontext an, indem Sie die neuen Gruppen und Gadgetlisten verwenden wollen.

Eine Gadgetspalte

Dieses Beispiel zeigt eine universell einsetzbare Gruppe, die eine beliebige Anzahl von Geometrien in einer Spalte anordnet. Die Elemente haben dabei alle den gleichen Abstand zum linken und rechten Rand der Gruppe.

```
class GadgetColumnC : public GeometryGroupC {
public:
   GadgetColumnC(
      GroupLayoutC &frame,
      ULONG topRelation, GeometryC *topObject, WORD topOffset,
      ULONG bottomRelation, GeometryC *bottomObject, WORD bottomOffset,
      ULONG leftRelation, GeometryC *leftObject, WORD leftOffset,
      ULONG rightRelation, GeometryC *rightObject, WORD rightOffset);
```

Zu der Gruppe werden die gewünschten Geometrien addiert. Diese Geometrien können mit beliebigen Regeln erzeugt sein, da die Methode `add` die Regeln geeignet überschreibt (s.u.)

```
   VOID add(GeometryC &);
private:
```

Die transparenten Geometrien werden benötigt, um den Platz für den rechten und linken Rand zu berechnen.

```
   TransparentLayoutC left, right;
   GeometryC leftG, rightG;
   GeometryC *lastG;
};

GadgetColumnC::GadgetColumnC(
      GroupLayoutC &frame,
      ULONG topRelation, GeometryC *topObject, WORD topOffset,
      ULONG bottomRelation, GeometryC *bottomObject,
      WORD bottomOffset,
      ULONG leftRelation, GeometryC *leftObject, WORD leftOffset,
      ULONG rightRelation, GeometryC *rightObject, WORD rightOffset)
   : GeometryGroupC(frame,
      topRelation,topObject,topOffset,
      bottomRelation,bottomObject,bottomOffset,
      leftRelation,leftObject,leftOffset,
      rightRelation,rightObject,rightOffset),
      left(),
      right(),
```

Zur Berechnung des linken Randes wird ein transparentes Element verwendet, dessen Geometrie links an die Gruppe gehängt wird und durch die Regel der rechten Koordinate eben so groß gemacht wird, wie das Maximum der linken Ränder alle Geometrien in der Gruppe.

```
leftG(left,
LAYOUT_SIZE,NULL,0,LAYOUT_GROUP,NULL,0,
LAYOUT_GROUP,NULL,0,LAYOUT_MAXOPPBORDER,NULL,0),
```

Für den rechten Rand wird entsprechend die Geometrie an den linken Gruppenrand gehängt und das Element so breit gemacht, wie die maximale Breite für die rechten Ränder angibt.

```
rightG(right,
LAYOUT_SIZE,NULL,0,LAYOUT_GROUP,NULL,0,
LAYOUT_MAXOPPBORDER,NULL,0,LAYOUT_GROUP,NULL,0),
lastG(NULL)
{
  GeometryGroupC::add(leftG);
  GeometryGroupC::add(rightG);
}

VOID GadgetColumnC::add(GeometryC &g)
{
```

Falls schon ein Element in die Liste eingehängt wurde, wird das neue Element an den unteren Rand dieses Elements angehängt.

```
  if (lastG)
    g.setTopRule(LAYOUT_OPPBORDER,lastG,2)
  else
```

Das erste Element wird allerdings an den oberen Rand der Gruppe angehängt.

```
    g.setTopRule(LAYOUT_GROUP,NULL,0);
```

Jedes Element ist so hoch, wie es sein Element angibt.

```
  g.setBottomRule(LAYOUT_SIZE,NULL,0);
```

Das Element wird links und rechts an die jeweiligen transparenten Elemente angehängt. Daß für den Rand des Elements genügend Platz ist, garantieren die Geometrien der transparenten Elemente durch die Verwendung der Regel **LAYOUT_MAXOPPBORDER**.

```
  g.setLeftRule(LAYOUT_OPP,&leftG,0);
  g.setRightRule(LAYOUT_OPP,&rightG,0);
  GeometryGroupC::add(g);
  lastG = &g;
}
```

Diese Gruppe kann nun allgemein eingesetzt werden, eine entsprechende Gruppe für Zeilen von Gadgets zu erstellen sollte kein Problem sein. Es müssen nur die entsprechenden Regeln vertauscht werden. In der Demo-Schublade liegt ein Beispiel für die Gadgetspalte bereit.

Die Grenzen des Layouters

Der Layouter kann nicht jede theoretisch denkbare Kombination aller Regeln korrekt berechnen - eine Implementationseinschränkung betrifft die beiden Prozentregeln:

- Wenn die Größe der Fläche, in die das Layout passen soll, bekannt ist, d.h. in der Layoutanweisung angegeben wird (z.B. eine vorgegebene Größe eines Fensters), so funktionieren diese Regeln einwandfrei. Soll jedoch die notwendige Größe der Fläche erst berechnet werden (und das geschieht in der Klasse `LayouterWindowC` vor dem Öffnen des Fensters), so kommen die Prozentregeln an ihre Grenzen, denn eine Prozentangabe gibt nunmal keine notwendige Größe der umgebenden Fläche vor.

Die Prozentregeln versuchen dennoch ihr bestes:

- Die umgebende Fläche wird wenigstens so groß gemacht, daß das Element mit der Prozentregel ganz hineinpasst. Wenn also dieses Element direkten Kontakt mit der umgebenden Gruppe hat (durch die Regel `LAYOUT_GROUP` in der anderen Koordinate), so kann die notwendige Fläche berechnet werden. Ebenso kann man bei den Prozentregeln ein Element angeben, neben dem das Element mit der Prozentregel ohne Überlappung liegen soll - das obige Beispiel "Eine Knopfzeile" verwendet diese Möglichkeit, um genügend Platz in der Zeile zu erhalten.

Es ist also sinnvoll, die Prozentregeln immer von einem Rand ausgehend anzugeben (d.h. die Elemente mit der Prozentregel sollten entweder direkt Kontakt mit dem Gruppenrand oder einem Element mit einer Prozentregel haben); um dies zu erreichen muß man unter Umständen die Elemente in geeignete Gruppen zusammenfassen.

Die anderen Regeln haben diese Schwierigkeiten in der Bestimmung des benötigten Platzes nicht.

Zur Gestaltung der Oberfläche

Der Autor will hier natürlich keine neuen Richtlinien für das Aussehen oder gar Verhalten der Oberfläche geben. Allerdings wird man feststellen, das man die Oberflächen, mit den Elementen, die durch diese Bibliothek zur Verfügung stehen, auf verschiedene Arten graphisch gestalten kann.

Erstens der Einsatz von Rahmen zur graphischen Gruppierung der Elemente. Dazu steht eine Klasse `LBBevelboxC` zur Verfügung, die Rahmen mit Titel zeichnet. Ab Version 39 des Betriebssystems sind dies normalerweise in die Oberfläche eingeritzte Rahmen , unter älteren Version die einfachen eingesenkten oder hervorstehenden Rahmen.

Die zweite Möglichkeit verwendet Patterns, zur farblichen Gruppierung der Elemente. Dies wird ebenfalls durch die Klasse `LBBevelboxC` unterstützt, hierbei sollte man jedoch als Rahmenart den

einfachen eingesenkten Rahmen verwenden (der einzige Rahmen der unter Version 37 und 38 zur Verfügung steht).

Die dritte Möglichkeit ist die Gruppierung durch horizontale und vertikale Striche, die die Klasse `LBBevellineC` zur Verfügung stellt.

Selbstverständlich können diese Möglichkeiten auch kombiniert werden (in manchen Fällen läßt sich das auch kaum vermeiden), jedoch sollte man damit sparsam umgehen - der Benutzer könnte durch eine zu musterreiche und zu stark unterteilte Oberfläche eher verwirrt werden, als dadurch Orientierung zu erhalten.

Die Fensterklasse LayouterWindowC

Die Klasse `LayouterWindowC` erbt von der Klasse `GTWindowC` und stellt Fenster zur Verfügung, deren Gadgets durch den Layouter verwaltet werden. Die Größe des Fensters wird beim Öffnen automatisch berechnet und kann durch den Benutzer verändert werden. Im Konstruktor wird ein Zeichensatz angegeben, der für alle Gadgets verwendet wird, er kann jederzeit verändert werden.

Die Fensterfläche ist in 5 Gebiete unterteilt: die innere Fläche (in der die meisten Gadgets liegen) und die 4 Ränder des Fensters.

In den unteren und rechten Rand werden häufig Rollgadgets gelegt, um den sichtbaren Teilbereich eines dargestellten Dokuments zu verschieben. Die folgende Klasse zeigt, wie diese Gadgets in den Rand gelegt werden:

```
class ScrollerWindowC : public LayouterWindowC {
public:
   ScrollerWindowC(GTIDCMPortC &, ScreenC &);
   ~ScrollerWindowC();
   TextAttrC font;
   LBScrollerC vertical,horizontal;
   GeometryC verticalGeo,horizontalGeo;
};

ScrollerWindowC::ScrollerWindowC(GTIDCMPortC &p, ScreenC &s)
   : LayouterWindowC(p,s,font,
     WA_SizeBRight,TRUE,
     WA_SizeBBottom,TRUE,
     WA_CloseGadget,TRUE,
     WA_DepthGadget,TRUE,
     WA_SizeGadget,TRUE,
     WA_DragBar,TRUE,
     TAG_END),
     font("topaz.font",8),
     vertical(NULL,*this,0,LAYOUT_AUTOSIZE,
      GA_RightBorder,TRUE,
      PGA_Borderless,TRUE,
      PGA_Freedom,FREEVERT,
      PGA_NewLook,TRUE,
      TAG_END),
```

```
        horizontal(NULL,*this,LAYOUT_AUTOSIZE,0,
         GA_BottomBorder,TRUE,
         PGA_Borderless,TRUE,
         PGA_Freedom,FREEHORIZ,
         PGA_NewLook,TRUE,
         TAG_END),
        verticalGeo(vertical,
         LAYOUT_GROUP,NULL,0,LAYOUT_GROUP,NULL,-2,
         LAYOUT_GROUP,NULL,2,LAYOUT_GROUP,NULL,-2),
        horizontalGeo(horizontal,
         LAYOUT_GROUP,NULL,1,LAYOUT_GROUP,NULL,-1,
         LAYOUT_GROUP,NULL,0,LAYOUT_GROUP,NULL,-2)
{
```

Die Gadgets werden in 3 Listen gespeichert: Titel-, BOOPSI- und GadTools-Gadgets. Die BOOPSI- und GadTools-Gadgets erhalten immer den gleichen Zeichensatz, die Titel-Gadgets (die natürlich normalerweise auch BOOPSI-Gadgets sind) sind zur graphischen Gruppierung der anderen Gadgets gedacht und können einen eigenen Zeichensatz erhalten. Die 3 Gadgetlisten heißen **gadgets**, **gtgadgets** und **titlegadgets**.

```
    gadgets.add(vertical);
    gadgets.add(horizontal);
```

Die 5 Geometriegruppen heißen **innerGeo** für die innere Fläche; **topGeo**, **bottomGeo**, **leftGeo** und **rightGeo** für die 4 Ränder.

```
    rightGeo.add(verticalGeo);
    bottomGeo.add(horizontalGeo);
}

ScrollerWindowC::~ScrollerWindowC()
{
  close();
}
```

Die Zeichensätze der Gadgets werden als Pointer angegeben, und müssen deshalb gültig sein, solange das Fenster geöffnet ist.

Die Fensterklasse StandardWindowC

Die Klasse **StandardWindowC** erbt von der Klasse **LayouterWindowC** und vereinfacht deren Gebrauch. Die Angabe der Zeichensätze geschieht zwar weiterhin über Pointer auf Objekte der Klasse **TextAttrC**, jedoch werden diese Zeichensatzbeschreibungen kopiert.

Außerdem kann im Konstruktor der Klasse **StandardWindowC** für ein Fenster ein Default-Gadget angegeben werden. Dieses Gadget erhält einen eigenen Zeichensatz, der immer **fett** eingestellt ist,

aber in Größe und Familie mit dem normalen Zeichensatz übereinstimmt. Dieser fette Zeichensatz kann auch noch für andere Zwecke verwendet werden.

Beim ersten Öffnen des Fensters wird automatisch der Standardzeichensatz des Bildschirms, auf dem das Fenster geöffnet wird, für beide Zeichensätze des Fenster verwendet.

Datei- und Verzeichnissperren

Wenn man auf den Verzeichniseintrag einer Datei, und nicht deren Inhalt zugreifen möchte, z.B. um zu testen, ob eine bestimmte Datei existiert, sollte man Dateisperren (Locks) verwenden.

Die Klasse `FileLockC` ermöglicht den einfachen Umgang mit den Dateisperren. Es stehen in dieser Klasse alle Funktionen der DOS-Library, die auf Dateisperren operieren, zur Verfügung. Einige Methoden benötigen weitere Strukturen der DOS-Library, z.B. die Methode `FileLockC::examine` benötigt einen `FileInfoBlock`, auf dem sie bestimmte Informationen über eine Datei oder ein Verzeichnis ablegt. Diese Methoden existieren immer in zwei Varianten:

- Eine Variante erwartet einen Pointer auf die Struktur der DOS-Library, eine andere Variante erwartet eine Referenz auf ein Objekt einer entsprechenden Klasse (z.B. `FileInfoBlockC` zur Struktur `FileInfoBlock`), die die Verwaltung dieser Struktur für sie übernimmt (insbesondere Allozierung und Freigabe).

Die Klasse `FileLockC` hat einen Erben: die Klasse `DirLockC`. Diese Klasse erweitert die Klasse `FileLockC` um Methoden, die typisch für Operationen auf Verzeichnissen sind, insbesondere natürlich das Einrichten eines neuen Verzeichnisses (während neue Dateien durch Objekte der Klasse `FileHandleC` erzeugt werden können). Desweiteren enthält diese Klasse Methoden zum „Scannen" eines Verzeichnisses.

Bis auf das Programm "*cd*" darf kein Programm das aktuelle Verzeichnis des CLI ändern. Wenn Sie dennoch innerhalb ihres Programms das aktuelle Verzeichnis ändern wollen, müssen Sie es am Ende des Programms unbedingt wieder auf den Anfangswert zurücksetzen. Die Klasse `CurrentDirC` erleichtert Ihnen diese Arbeit, indem bei der Konstruktion eines Objekts dieser Klasse das Objekt das aktuelle Verzeichnis speichert, und bei der Destruktion das aktuelle Verzeichnis wieder auf das Gespeicherte zurücksetzt.

Wenn Sie also vorhaben, in ihrem Programm das aktuelle Verzeichnis zu ändern, sollten Sie einfach am Anfang der `main()`-Funktion oder `wbmain()`-Funktion ein Objekt dieser Klasse deklarieren (eine globale Variable des Typs `CurrentDirC` bringt ebenfalls den gewünschten Effekt).

Die Dateiklassen

Die eigentliche Dateiklasse `FileHandleC` der Bibliothek basiert direkt auf den Funktionen der DOS-Library.

Die Lese- und Schreibmethoden (`FileHandleC::read` und `FileHandleC::write`) operieren wie die Originalfunktionen `Read()` und `Write()` auf typenlosen Puffern. Hier stellt jedoch die Klassenbibliothek einen geeigneten dynamischen Puffer zur Verfügung, den Sie zum Lesen und Schreiben einer Datei verwenden können. Die Klasse `BufferC`.

Sie können die Puffergröße eines Objekts der Klasse `BufferC` jederzeit ändern, müssen jedoch beachten, daß sich dadurch natürlich die Adresse des Puffers ändert. Sie dürfen daher die Adresse nicht in einer Variablen zwischenspeichern, sondern sollten diese Adresse bei jeder Verwendung des Puffers neu ermitteln.

Das Kopieren einer Datei könnte dann ungefähr so aussehen:

```
FileHandleC von, nach;
if (von.open(VON_NAME, MODE_OLDFILE) && nach.open(NACH_NAME, MODE_NEWFILE))
{
   BufferC puffer(10000);
   ULONG i;
   while ((i = von.read(puffer.buffer(),puffer.size())) > 0)
   {
      if (nach.write(puffer.puffer(),i) < 0)
        break;
   };
};
```

Die Dateien werden natürlich im Destruktor der Klasse `FileHandleC` automatisch geschlossen.

Von der Klasse `FileHandleC` erben zwei weitere Klassen: die Klasse `InputHandleC` für die Standardeingabe und die Klasse `OutputHandleC` für die Standardausgabe. Im Normalfall ist es jedoch sinnvoller, die Objekte `cin` und `cout` aus der Header-Datei *<iostream.h>* für diesen Zweck zu verwenden, nur wenn Sie globale Variablen in Ihrem Programm komplett vermeiden wollen, sollten Sie auf die Klassen `InputHandleC` und `OutputHandleC` zurückgreifen.

Kommandozeilenargumente

Die Klassenbibliothek vereinfacht die Auswertung der Kommandozeilenargumente stark. Sie müssen nur eine Variable des Typs `ArgsC` deklarieren, dem Konstruktor dieser Klasse übergeben Sie das Template der Argumente.

Danach können Sie die verschiedenen Argumentenwerte direkt ermitteln, entweder durch Angabe des Argumentennamens, den Sie im Template angegeben haben, oder der Nummer des Arguments. Die Klasse `ArgsC` unterscheidet dabei String-Argumente, Zahl-Argumente und Schalter-Argumente.

Ebenso einfach ist es, Mehrfachargumente auszuwerten: Die Klasse `ArgsMCursorC` stellt Objekte zur Verfügung, deren Konstruktor Sie wiederum die Nummer oder den Namen des Mehrfacharguments übergeben müssen. Dann können Sie auf die Werte des Arguments durch die typischen Methoden eines Cursors zugreifen.

Beschreibung des HotHelp Projekts zu EASY-OBJECTS

Das Handbuch zu EASY-OBJECTS beschreibt die Grundkonzepte und Verwendung der Klassenbibliothek, für die täglich Arbeit steht Ihnen eine ausführliche Hypertextdokumentation zur Verfügung. Das HotHelp-Projekt EASY-OBJECTS, das bei der Installation der Klassenbibliothek ebenfalls installiert wird.

Bevor Sie die folgende Beschreibung des HotHelp-Projekts EASY-OBJECTS lesen, sollten Sie sich mit der Bedienung von HotHelp vertraut gemacht haben.

Öffnen Sie das HotHelp-Fenster (üblicherweise durch den Tastendruck Alt-Help) und wählen Sie die Projektliste aus (Knopf "Projekte"). Es erscheint die Startseite des Projekts EASY-OBJECTS.

Unter der Titelgraphik befinden sich die Referenzen des Inhaltsverzeichnisses:

Einführung

Diese Referenz führt Sie zu einem Beispiel, wie EASY-OBJECTS benutzt wird; es wird eine besondere Version des unvermeidlichen Programms "HelloWorld" vorgestellt. Dieses Kapitel befindet sich auch im Handbuch.

Hintergrund

Diese Referenz führt Sie zu einem weiteren Inhaltsverzeichnis, in dem Sie die restlichen Kapitel des Handbuchs (bis auf diese Beschreibung natürlich) finden. Sie haben also die freie Wahl, ob Sie die Verwendung der Bibliothek lieber am Monitor, oder aus dem Handbuch lernen wollen...

Diese Kapitel enthalten ausführlich dokumentierte Beispiele, die den jeweiligen konzeptionellen oder technischen Aspekt der Klassenbibliothek vorstellen. Die Beispiele liegen selbstverständlich auch als lauffähige Programme in der Demo-Schublade von MaxonC++ vor.

Technische Kapitel

Dieser Überschrift folgt einer Referenzenliste, jede Referenz verweist auf ein Inhaltsverzeichnis eines Teilbereichs der Klassenbibliothek. Die Einteilung folgt der Einteilung der Betriebssystemdokumentation, z.B. Exec, DOS, Intuition etc. In diesen Kapiteln finden Sie die Beschreibung der Klassen, die Ressourcen aus den entsprechenden Bibliotheken des Betriebssystems verwalten, also Klassen zur Speicherverwaltung, zu Signalen und Nachrichtenports in Exec, Klassen zu Dateien und Locks in DOS und Klassen zur Fensterverwaltung und Gadgets in Intuition.

Drei Kapitel fallen dabei aus der Reihe:

Das Kapitel Boopsi

ist aus Intuition ausgegliedert und beschreibt die Klassen des Boopsi-Systems. Darin befinden sich sowohl die BOOPSI-Klassen des Betriebssystems, wie auch neue BOOPSI-Gadgets und Images.

Das Kapitel DataStructures

beschreibt Klassen, die verschiedene Datenstrukturen zur Verfügung stellen: dynamische Arrays, dynamische Datenpuffer, Stringverwaltung usw.

Das Kapitel Layouter

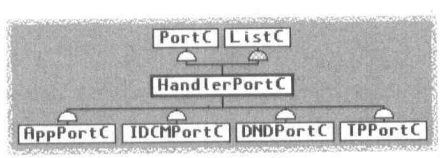
beschreibt die Klassen des Layouters, mit dem Sie zeichensatzunabhängige und größenveränderbare Bedienoberflächen erzeugen können.

Die Namen aller Kapitel stimmen übrigens mit dem Namen des Verzeichnisses überein, in dem die Header-Dateien des jeweiligen Kapitels liegen.

Alle technischen Kapitel besitzen einen identischen Aufbau:

Das Inhaltsverzeichnis des Kapitels führt sämtliche Klassennamen dieses Teilbereichs als Referenzen auf. Jede Referenz führt zu der Beschreibung einer Klasse.

Zuerst wird kurz die Bedeutung der Klasse beschrieben, darauf folgt eine Graphik, in der Sie erkennen können, welche Klassen diese Klasse beerbt, und an welche wichtigen Klassen diese Klasse vererbt.



Die Abbildung zeigt den Klassenbaum der Klasse **HandlerPortC**, der Name dieser Klasse steht in der mittleren Zeile und ist dick umrandet. Darüber stehen die beiden Klassen, die **HandlerPortC** beerbt, **PortC** und **ListC**. In der unteren Zeile stehen zwei Klassen, an die **HandlerPortC** erbt, **AppPortC** und **IDCMPortC**. Es folgt der Name der Header-Datei, in der die Definition der Klasse steht.

Wenn Sie eine Header-Datei einbinden wollen, so sollten Sie aus Geschwindigkeitsgründen die Include-Anweisung mit einem `#ifdef` ... `#endif` klammern. Das Symbol, auf dessen Existenz Sie in der `#ifdef`-Anweisung testen, setzt sich dabei aus dem Namen der Header-Datei zusammen.

Alle Header-Dateien liegen im Verzeichnis "*Classes*" im Include-Pfad des Compilers, entsprechend beginnt das Symbol mit"`CPP_`". Der zweite Teil des Symbols ist der großgeschriebene Name des Verzeichnisses in dem die Header-Datei liegt, der dritte Teile ist der Name der Datei, wobei das Suffix "*.h*" durch "*_H*" ersetzt und außerdem alles groß geschrieben wird. Verständlicher als viele Worte sind wohl einige Beispiele:

Header-Datei: Classes/Exec/Lists.h ---> `CPP_EXEC_LISTS_H`

Header-Datei: Classes/Layouter/Windows.h ---> `CPP_LAYOUTER_WINDOWS_H`

Eine vollständige Include-Anweisung sollte dann folgendermaßen aussehen:

```
#ifndef CPP_EXEC_LISTS_H
#include <classes/exec/lists.h>
#endif
```

Als nächstes folgt die C++ Definition der Klasse. In dieser Defintion sind natürlich auch alle anderen Klassennamen referenziert, insbesondere können Sie darüber auch die Beschreibung zu den Elternklassen lesen.

Ein Tip

HotHelp kann Textteile exportieren, z.B. ins Clipboard legen oder direkt in den Editor einfügen. Wenn Sie eine Klasse ableiten, und wollen Methoden überladen, können Sie die benötigten Prototypen direkt aus dem HotHelp Projekt in Ihren C++ Quelltext exportieren.

Dieser Klassendefintion folgt eine Zeile in der alle abgeleiteten Klassen (nicht nur die wichtigen Klassen aus der Graphik) aufgeführt sind. Damit sind Sie in der Lage, den gesamten Klassenbaum abzulaufen. Gute Dienste leistet Ihnen dabei der Knopf "***Vortext***" von HotHelp, durch den Sie auf einen Vorgängertext zurückblättern.

Der Rest jeder Seite enthält die Beschreibungen aller wichtigen Konstruktoren und Methoden. Diese werden mit dem Prototyp und einer kurzen Beschreibung dargestellt.

Jede Methode besitzt einen unsichtbaren Schlüssel, der sich aus dem qualifizierten Methodennamen zusammensetzt (z.B. `PortC::handleMsg`, also erst der Klassenname, dann zwei Doppelpunkte und schließlich der Methodenname). In der Beschreibung tauchen immer wieder Referenzen auf diese unsichtbaren Schlüssel auf, die Sie unmittelbar zur Beschreibung einer Methode führen - z.B. wenn ein Methode einer Klasse eine Methode einer Elternklasse überlädt.

Benutzung des HotHelp Projekts EASY-OBJECTS

Die Klassenbibliothek enthält über 300 verschiedene Klassen. Selbstverständlich können Sie nicht alle Klassen auswendig kennen, darum werden Sie ständig auf das HotHelp Projekt zurückgreifen müssen, um die Namen der Klassen oder auch Prototypen der Methoden nachzuschlagen.

Hier folgt nun ein Beispiel, wie Sie beim Aufbau eines Programms Hilfe durch das HotHelp Projekt bekommen.

Das Programm soll ein Fenster mit einem Knopf darin öffnen. Wenn man den Knopf drückt, soll auf der Standardausgabe der Text **"Knopf gedrückt"** ausgegeben werden. Das Programm soll entweder durch das Schließsymbol des Fensters oder durch <Ctrl>+<C> beendet werden können.

Wenn Sie nicht wissen, wie die geeignete Fensterklasse überhaupt heißt, dann öffnen Sie das HotHelp Fenster und gehen Sie in das Projekt **"EASY-OBJECTS"**. Für Bedienoberflächen sollte man immer den Layouter verwenden, also gehen Sie in das Kapitel **"Layouter"**. Dort finden sich zwei Einträge zu Fenstern: **LayouterWindowC** und **StandardWindowC**. Lesen Sie sich beide Beschreibungen der Klassen durch, dann werden Sie feststellen, daß die Klasse **StandardWindowC** die geeignete Klasse ist.

Also leiten wir eine Klasse von dieser Klasse ab:

```
class KnopfWindowC : public StandardWindowC {
```

Der Konstruktor muß natürlich die nötigen Parameter enthalten, die auch die Elternklasse benötigt.

```
public:
   KnopfWindowC(GTIDCMPortC &, ScreenC &);
   ~KnopfWindowC();
```

Nun wollen wir einen Knopf für den Layouter verwenden. Auch hier können wir wieder in HotHelp im Kapitel **"Layouter"** nachschauen. Dort finden sich einige Gadgets, die Knöpfe sind: ***LBButtonC***, ***LBFrButtonC***, ***LBIButtonC*** und ***LGTButtonC***. Die Namenskonvention von EASY-OBJECTS ist dabei folgende: das "*L*" am Anfang der Namen steht für den Einsatz der Klasse im Layouter, das folgende "*B*" für BOOPSI-Gadgets, oder das folgende "*GT*" für GadTools-Gadgets. Die beiden Klassen `LBButtonC` und `LBFrButtonC` sind, wie Sie feststellen können, wenn Sie die Klassen bis zu ihren Elternklassen `BButtonC` und `BFrButtonC` verfolgen, die C++ Klassen zu den System BOOPSI Klassen `BUTTONGCLASS` und `FRBUTTONCLASS`. Beide sind nicht sonderlich einfach zu verwenden, die Klasse `LBIButtonC` ist ein BOOPSI-Knopf, der deutlich besser den Layouter unterstützt. Wählen wir also diesen Knopf (der GadTools-Knopf ginge genauso).

Anhand des Konstruktors sehen wir, daß ein Knopf der Klasse `LBIButtonC` wie jedes andere Gadget auch, eine Instanz der Klasse `GadgetEventC` benötigt, die die Reaktion auf die Betätigung durch den Benutzer festlegt. Dafür deklarieren wir eine Klasse `KnopfEventC`, die wir später noch genauer bestimmen.

Zudem benötigt jedes Element im Layouter eine Geometrie, diese deklarieren wir auch noch.

```
private:
   KnopfEventC knopfEvent;
   LBIButtonC knopf;
   GeometryC knopfGeo;
```

Um das Programm durch das Schließsymbol des Fensters zu beenden, benötigen wir einen geeigneten Handler, der die IDCMP Nachricht verarbeitet. Im Kapitel Intuition findet sich bei den Fensterklassen eine Klasse namens `WindowCloseHandlerC` - klingt eigentlich ganz brauchbar. Die Beschreibung dieser Klasse bzw. der Methode `WindowCloseHandlerC::close()` erfüllt genau unsere Wünsche.

```
   WindowCloseHandlerC wch;
};
```

Die Implementation des Konstruktors ergibt sich automatisch aus den Konstruktoren der ausgewählten Klassen. Denken Sie daran, daß Sie mit Shift-Help direkt das Wort unter dem Cursor in HotHelp suchen können. Stellen Sie den Cursor einfach auf den Namen einer verwendeten Klasse, um den Prototyp des Konstruktors oder einer Methode nachzulesen.

```
KnopfWindowC::KnopfWindowC(GTIDCMPortC &p, ScreenC &s)
   : StandardWindowC(p,s,NULL,
     TAG_END),
```

Um zur Beschreibung der BOOPSI-Klasse für den Knopf zu kommen, muß man 2 Referenzen durchlaufen: Die Elternklasse `BIButtonC` der Klasse `LBIButtonC` nennt den Namen der BOOPSI-Klasse: `IButtonClass`. Dort finden sich alle neuen Attribute dieser Klasse. Für unseren Zweck reichen jedoch die Standardattribute `GA_Text` (das immer für Beschriftungstexte verwendet wird) und `GA_RelVerify` (das immer auf `TRUE` gesetzt werden muß, damit Intuition die Betätigung des Gadgets an das Programm weitergibt).

```
     knopf(&knopfEvent,*this,LAYOUT_AUTOSIZE,LAYOUT_AUTOSIZE,
      GA_Text,"Ein Knopf",
      GA_RelVerify,TRUE,
      TAG_END),
```

Die Beschreibung der Regeln, die Sie im Konstruktor der Geometrie angeben, finden Sie auch in HotHelp unter dem Schlüssel **"Die Regeln"**.

```
     knopfGeo(knopf,
      LAYOUT_GROUP,NULL,2,LAYOUT_GROUP,NULL,-2,
      LAYOUT_GROUP,NULL,2,LAYOUT_GROUP,NULL,-2),
     wch(*this)
{  gadgets.add(knopf);
   innerGeo.add(knopfGeo);
}
```

```
KnopfWindowC::~KnopfWindowC()
{
  close();
}
```

Die Klasse `KnopfEventC`, die wir im Fenster verwenden, muß noch definiert werden. Sie wird von der Klasse `GadgetEventC` abgeleitet (dies konnten wir an der Defintion des Konstruktors der Gadgetklasse `LBIButtonC` sehen).

```
class KnopfEventC : public GadgetEventC {
public:
  KnopfEventC() : GadgetEventC() { };
```

Diese Methode wird aufgerufen, wenn das Gadget gedrückt und wieder losgelassen wurde. Den Prototyp können Sie einfach aus HotHelp exportieren. Da nur eine einfache Ausgabe vorgenommen wird, ist die Methode inline deklariert.

```
  VOID up(WindowC *, GadgetC *, IntuiMessageC * )
    { cout << "Knopf gedrückt.\n"; };
};
```

Das Hauptprogramm soll hier nicht mehr im Einzelnen beschrieben werden, es sieht in allen Programmen gleich aus.

Natürlich müssen die verschiedenen Header-Dateien der Klassen, die im Programm benutzt werden, geladen werden. Welche Dateien Sie benötigen, können Sie auch in HotHelp nachlesen, der Name steht über der jeweiligen Klassendefinition.

Hier folgt nun der vollständige (und richtig sortierte) Quelltext des Beispiels, das natürlich auch in der Demo-Schublade vorliegt.

```
#include <classes/Layouter/Windows.h>
#include <classes/Layouter/BoopsiGadgets.h>
#include <classes/Exec/Libraries.h>
#include <iostream.h>

class KnopfEventC : public GadgetEventC {
public:
  KnopfEventC() : GadgetEventC() { };
  VOID up(WindowC *, GadgetC *, IntuiMessageC * )
    { cout << "Knopf gedrückt.\n"; };
};

class KnopfWindowC : public StandardWindowC {
public:
  KnopfWindowC(GTIDCMPortC &, ScreenC &);
  ~KnopfWindowC();
private:
  KnopfEventC knopfEvent;
  LBIButtonC knopf;
  GeometryC knopfGeo;
  WindowCloseHandlerC wch;
```

```
};

KnopfWindowC::KnopfWindowC(GTIDCMPortC &p, ScreenC &s)
   : StandardWindowC(p,s,NULL,
     TAG_END),
     knopf(&knopfEvent,*this,LAYOUT_AUTOSIZE,LAYOUT_AUTOSIZE,
      GA_Text,"Ein Knopf",
      GA_RelVerify,TRUE,
      TAG_END),
     knopfGeo(knopf,
      LAYOUT_GROUP,NULL,2,LAYOUT_GROUP,NULL,-2,
      LAYOUT_GROUP,NULL,2,LAYOUT_GROUP,NULL,-2),
     wch(*this)
{
   gadgets.add(knopf);
   innerGeo.add(knopfGeo);
}

KnopfWindowC::~KnopfWindowC()
{
   close();
}

LibraryBaseErrC GadToolsBase("gadtools.library",37);
LibraryBaseErrC UtilityBase("utility.library",37);
LibraryBaseErrC CxBase("commodities.library",37);
LibraryBaseErrC LayersBase("layers.library",37);
LibraryBaseErrC WorkbenchBase("workbench.library",37);

int main()
{
   if (!LibraryBaseC::areAllOpen())
     return 20;

   SignalsC sc;

   PublicScreenC screen();
   if (!screen.lock(NULL))
     return 100;

   GTIDCMPortC port;
   sc.add(port);
   GadgetUpHandlerC uphandler;
   port.add(uphandler);
   KnopfWindowC window(port,screen);
   CtrlCHandlerC ctrlchandler;
   sc.add(ctrlchandler);
   window.open();
   sc.loop();

   return 0;
}
```

AMIGA

MaxonC++

Source-Level-Debugger

MAXON
computer

1. Einführung

Das Entwicklungssystem MaxonC++ 3 enthält einen völlig neu entwickelten Quellsprachen-Debugger. Der bisher vertriebene Debugger (des gleichen Names) wird nicht mehr unterstützt. Der Debugger erleichtert die Entwicklung funktionsfähiger Programme wesentlich, da mit ihm Fehler effektiv gesucht und gefunden werden können. Durch seine intuitive Bediendbarkeit und seine konsequente Hochsprachen-Orientierung ist er insbesondere für den Anwender von C++ geeignet.

Einige seiner Eigenschaften:

Mehrere Quelltexte gleichzeitig betrachtbar.

Darstellung aller Datenstrukturen in C bzw. C++ Syntax oder Pascal Syntax, es können sogar gemischte Projekte korrekt debuggt werden.

Läuft asynchron zum untersuchten Programm.

automatische Aktualisierung aller Variablenwerte.

konfigurierbare Oberfläche.

Was können Sie mit MaxonSDB tun?

MaxonSDB eignet sich für die Fehlersuche in Programmen, die mit dem Compiler MaxonC++ (ANSI C oder C++) ab Version 3 oder MaxonPASCAL erstellt sind. Sie können das Programm in einzelnen Schritten oder in größeren Sprüngen abarbeiten, dabei Variablenwerte beobachten und Datenstrukturen inspizieren. Programmierer, deren Rechner eine MMU besitzen, können den Enforcer verwenden um illegale Speicherzugriffe zu testen, mit dem Debugger können Sie direkt die entsprechende Sourcestelle zu einem Enforcerhit suchen. Bislang kann der Debugger keine disassemblierten Programmteile darstellen (bei der Entwicklung wurde wesentlich auf die Hochsprachen Rücksicht genommen) und nicht mehreren Prozesse oder Tasks gleichzeitig beobachten.

Über dieses Handbuch

In diesem Handbuch werden einige Begriffe in einer allgemeineren Bedeutung als üblich verwendet, und für manche Begriffe wird der englische Name genutzt, falls deren deutsche Übersetzung nicht gebräuchlich ist.

Mit **Modul** wird hier eine Übersetzungseinheit bezeichnet, wie sie C oder Assembler-Programmierern bekannt ist. Pascal benutzt hierfür den Begriff „Unit".

Der Begriff **Funktion** deckt sowohl die Funktionen der Sprache C, als auch die Unterprogramme und Prozeduren von Pascal und Assembler ab.

Argumente sind dasselbe wie **Parameter**. Es wird aber unterschieden zwischen den Kommandozeilenargumenten eines Programms und den Argumenten einer Funktion.

Kickstart: Das Betriebssystem der Amiga-Rechner wird oft unter verschiedenen Namen angesprochen, wobei diese Namen eigentlich nur für einen Teil des Betriebssystems stehen (z.B AmigaDOS 2.0 oder Workbench 1.3). In diesem Handbuch wird einheitlich der Name Kickstart für das Betriebssystem des Amiga verwendet. Dabei wird zwischen den Versionen 1.2 und 1.3 nicht unterschieden.

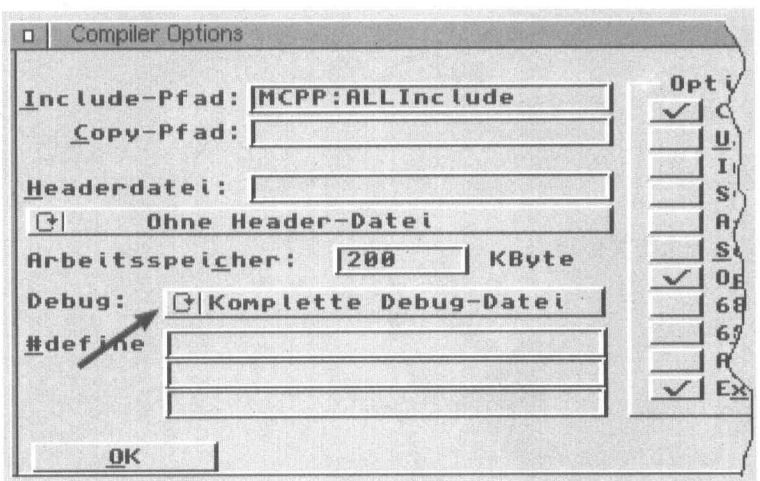
(Der) Debugger.

(Der) Startup-Code. Hierfür gibt es jeweils keinen gebräuchlichen deutschen Begriff.

2. Die Benutzung von MaxonSDB

Die Vorbereitung Ihres Programms

Damit Sie ihr Programm mit MaxonSDB debuggen können, muß ein Schalter in den Compiler-Optionen aktiviert werden. Nach dem Einschalten der Option ist es erforderlich, daß Sie das jeweilige Modul oder alle zum Programm gehörigen Module erneut compilieren! Bei der Kommandozeilenversion des MaxonC^{++} - Compilers muß bei jedem Aufruf die Option „-b" angegeben sein.



Andere Compiler erzeugen eventuell ebenfalls Debug-Dateien, die für MaxonSDB geeignet sind. Bitte entnehmen Sie die dafür nötigen Befehle und Schritte dem Handbuch des jeweiligen Compilers.

Programmstart

Sie können den Debugger auf drei verschiedene Arten starten: von der Workbench, aus der Shell oder direkt aus der Entwicklungsumgebung ihres Compilers.

Der Workbenchstart

Wie gewohnt startet der Doppelklick auf das Programmikon den Debugger. Mit dem Menüpunkt **„öffnen"** aus dem Menü Projekt können Sie die MDBG-Datei des zu debuggenden Programmes laden. Eine weitere Möglichkeit besteht darin, das Programm-Ikon des zu debuggenden Programmes in das Ikonfeld (AppIcon) im Projektfenster abzulegen.

Am einfachsten klicken Sie auf das vom Compiler erzeugte Programm-Ikon mit der Endung "*.mdbg*". Der Debugger wird dann automatisch gestartet und das zu debuggende Programm geladen und ebenfalls gestartet.

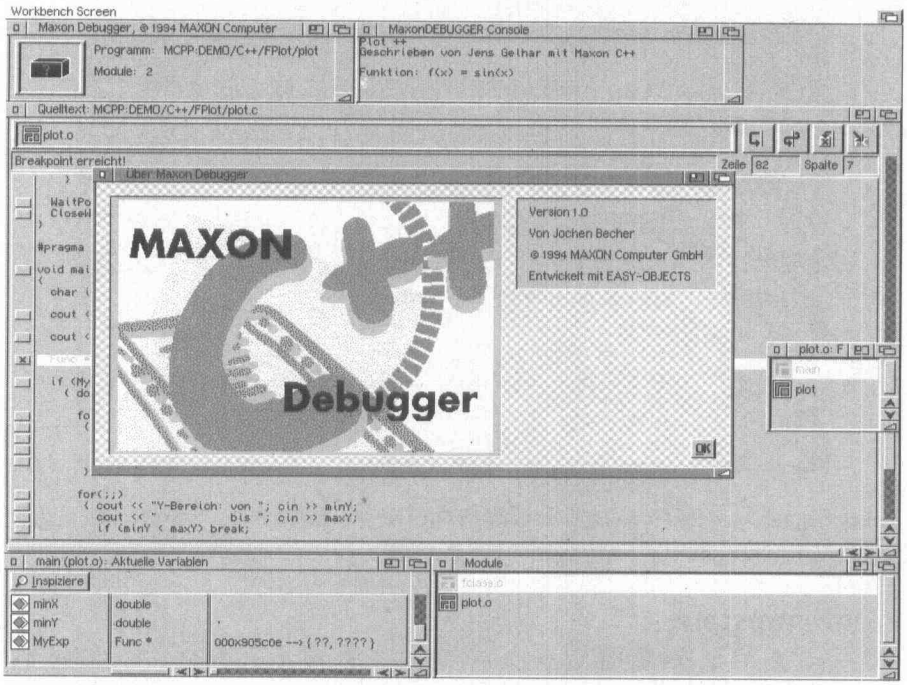
In den Tooltypes von MaxonSDB können folgende Einstellungen vorgenommen werden:

ARGUMENTS="Programmargumente"

Hier müssen die Argumente für das zu debuggende Programm angegeben werden. Die Argumente werden dann bei Programmstart übergeben.

PUBSCREEN=

Dem Tooltype PUBSCREEN können Sie den Namen eines öffentlichen Bildschirms angeben, auf dem alle Debugger-Fenster angezeigt werden sollen. Geben Sie dieses Argument nicht an, wird der Default-Bildschirm verwendet - üblicherweise die Workbench.



Der Shellstart

Starten Sie den Debugger durch Eingabe des Programmnamens MSDB. Als ein Debugger-Argument können Sie den Namen einer MDBG-Datei angeben oder einen Programmnamen. Dieses Programm wird dann in den Debugger geladen und gestartet. Beachten Sie allerdings, daß der Stack wenigstens 40000 Bytes groß sein muß.

Das vollständige Muster der Argumente sieht so aus: PROGRAM, ARGUMENTS, PUBSCREEN/K

Als zweites Argument können Sie die Argumente des zu untersuchenden Programms angeben (benutzen Sie immer Hochkommas für dieses Argument, wenn sie mehrere Argumente an das zu untersuchende Programm übergeben wollen). Das letzte Argument kann der Name eines öffentli-

chen Bildschirms sein, auf dem der Debugger seine Fenster öffnen soll. Geben Sie dieses Argument nicht an, wird der Default-Bildschirm verwendet - üblicherweise die Workbench.

Der Start aus der Entwicklungsumgebung

Dieser Start ist natürlich der einfachste. Kompilieren Sie ihr Programm mit der entsprechenden Compiler-Option um Debugdateien zu erstellen. Wurde Ihr Programm fehlerfrei übersetzt, können Sie den Debugger aus dem Menü **Compiler** starten. Ihr gerade übersetztes Programm wird dann automatisch in den Debugger geladen. Der Debugger übernimmt dabei die Argumente, die Sie im Dialog **Optionen/System** im Textfeld "***Argumente***" angegeben haben.

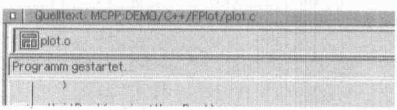
3. Die Fenster des Debuggers

Drag&Drop

In fast jedem Fenster des Debuggers besteht die Möglichkeit, das sogenannte Drag&Drop zu nutzen. Drag&Drop bedeutet ziehen und loslassen, womit schon ungefähr klar wird, was gemeint ist.

Wenn Sie mit der Workbench arbeiten, kennen Sie dieses Prinzip bereits. Es wird beim Kopieren, Verschieben und Installieren genutzt, wenn Sie mit der Maus ein Workbench-Ikon greifen und es über einem anderen Workbenchfenster loslassen.

In MaxonSDB haben viele Fenster Drag&Drop-Bereiche, die Sie daran erkennen können, daß zusätzlich zu einem namentlichen Eintrag ein Ikon angezeigt wird. Beim Klick auf den Namen wird der jeweilige Eintrag ausgewählt. Mit Klick auf das Ikon und Gedrückthalten der Maustaste können Sie das Ikon in den Drag&Drop-Bereich eines anderen Fensters verschieben.

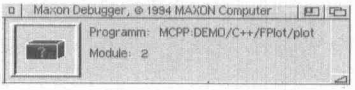


Versuchen Sie es doch gleich einmal. Laden Sie die MDBG-Datei des Programmes FPlot, das bereits mit fertigen Debug-Dateien vorliegt. Öffnen Sie nun ein Fenster für die aktuellen Variablen. Greifen Sie nun im Variablen-Fenster das Ikon des Eintrags `"input"` und ziehen Sie dieses Ikon in den Drag&Drop-Bereich des Quelltextfensters. Sobald Sie den Bereich mit der Maus berühren, wird er invertiert angezeigt und symbolisiert Ihnen so, daß das Ikon hier abgelegt werden kann. Sobald Sie das Ikon über dem invertierten Bereich loslassen wird die Quelltextstelle angezeigt, an der die lokale Variable definiert wird.

Wenn Sie aus dem Funktionen-Fenster Ikons auf den Drag&Drop-Bereich des Quelltextfensters verschieben, können Sie so sehr leicht die Quelltextposition von Funktionen auffinden und zum Beispiel Haltepunkte definieren.

Das Projektfenster

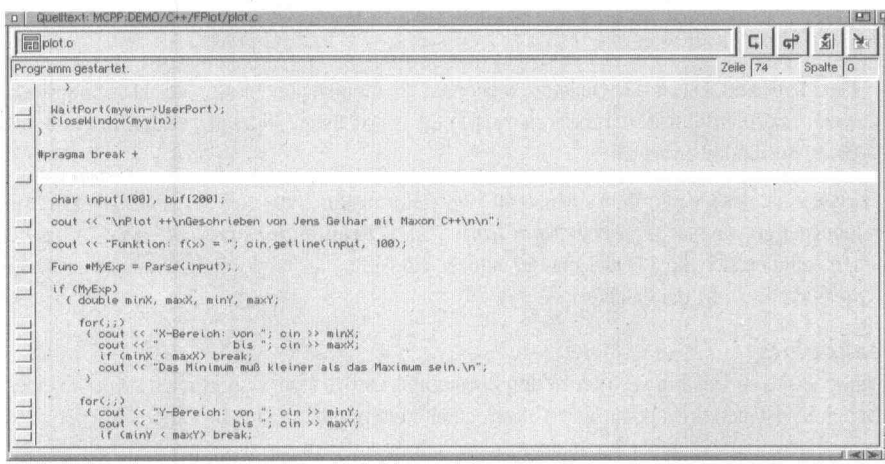
Nach dem Start wird grundsätzlich das Projektfenster des Debuggers geöffnet. Dieses enthält den Namen des geladenen Programms und die Anzahl der Module im Programm. Daneben befindet sich das Ikon des Programms, in dieses Feld können Sie auch ein Ikon der Workbench ziehen, um ein neues Programm zu laden.



Das Projektfenster ist das einzige Fenster, das immer geöffnet ist. Alle anderen Fenster können auch geschlossen sein - je nach Konfiguration ihrer Oberfläche haben Sie von den anderen Fenstern zu Anfang mehr oder weniger geöffnet. Die meisten Fenster können Sie aus dem Menü Fenster öffnen.

Das Quelltextfenster *<A 1>*

Sobald Sie ein Programm geladen und damit auch gestartet haben, wird zumindest ein Quelltextfenster geöffnet, in dem der Quelltext des Moduls mit der Main-Funktion angezeigt wird. Die aktuelle Zeile, d.h. die Zeile in der der PC (Program Counter) des Programms sich gerade befindet, wird invertiert dargestellt.



Das Quelltextfenster ist das wichtigste Fenster. Neben der Anzeige der aktuellen Position des Programms mit der invertierten Programmzeile sind darin auch die Knöpfe enthalten, mit denen der Ablauf des Programms gesteuert wird.

In der obersten Zeile befindet sich der Name des Moduls, dessen Quelltext gerade angezeigt wird. Daneben sind 4 kleine Knöpfe mit Bildsymbolen.

<A I> oder <I>

Mit dem ersten Knopf wird ein Einzelschritt ausgelöst. Das Programm läuft einen Schritt weiter und hält dann wieder an. Bei einem Unterprogrammaufruf wird in das Unterprogramm verzweigt und der entsprechende Quelltext dazu angezeigt.

Der verwendete Compiler bestimmt, wo ein Schritt beginnt und wo er endet. Schritte stimmen dabei mehr oder weniger mit den einzelnen Befehle überein - allerdings werden komplexe Ausdrücke, in denen Funktionsaufrufe stehen, in mehrere Schritte unterteilt.

<A R> oder <R>

Der zweite Knopf führt auch einen Einzelschritt aus, allerdings werden Unterprogramme, die in diesem Schritt aufgerufen werden immer schnell abgearbeitet. Unterprogramme, die nicht Teil des Programms sind (insbesondereBetriebssystemaufrufe) werden sowieso immer schnell abgearbeitet.

<A G> oder <G>

Der dritte Knopf läßt das Programm schnell weiter arbeiten. Dadurch läuft es entweder bis zum Ende durch, oder bis es auf einen Breakpoint stößt.

<A E> oder <E>

Der vierte Knopf setzt den Quelltext wieder auf die momentane Position des Programms, z.B. wenn man den Quelltext verschoben hat, oder sogar einen anderen geladen hat, um z.B. einen Breakpoint zu setzen.

Die zweite Zeile des Quelltextfensters enthält eine Statusmeldung, die darüber Aufschluß gibt, was der Debugger gerade macht (*„Programm läuft"*, oder *"Breakpoint erreicht"* oder ...). Rechts daneben befinden sich die 2 Felder mit der aktuellen Zeile und Spalte des Programms im angezeigten Quelltext. Nun folgt der Quelltext.

Breakpoints

Die linke Spalte im Quelltext-Fenster ist den Breakpoints vorbehalten. Dort können sie durch einen einfachen Klick einen Breakpoint setzen, durch einen weiteren Klick einen Breakpoint löschen. Den gesetzen Breakpoint erkennen Sie an einem kleinen Kreuz auf dem jeweiligen Knopf.

Eine zweite Art von Breakpoints setzen Sie, wenn Sie den Klick auf einen Breakpointknopf bei gedrückter Shifttaste durchführen. Statt einem Kreuz erscheint ein schwarzer Punkt. Dieser Breakpoint wird beim Erreichen automatisch gelöscht.

Den Quelltext können Sie mit den beiden Scrollern sowohl vertikal, wie horizontal verschieben.

Das Modulfenster <A 2>

Das Modulfenster zeigt die Namen aller geladenen Module an. Dieses Fenster besteht aus einer Liste der Modulnamen. Links neben jedem Namen befindet sich ein kleines Ikon, daß Ihnen für Drag & Drop zur Verfügung steht.

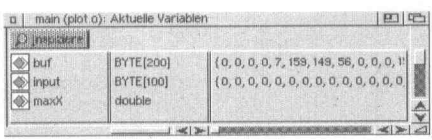
Jedes Programm hat sicher mehrere Module, und die meisten Benutzer (insbesondere die glücklichen Besitzer von Graphikkarten und großen Monitoren) werden auch mehrere Quelltextfenster öffnen. Der einfachste Weg, den Quelltext eines Moduls in einem bestimmten Quelltextfenster anzuzeigen besteht in der Verwendung des Drap&Drop Prinzips: Greifen Sie das Ikon eines Moduls (wie Sie es auch von der Workbench gewohnt sind) und verschieben Sie es auf die erste Zeile eines Quelltextfensters - eben dort wo der Namen des bisher geladenen Modules steht.

Sie werden sehen, daß sich die Fläche dieses Namens invertiert - zum Zeichen, daß dieses Oberflächenelement ein Ikon empfangen kann. Lassen Sie einfach das Ikon los und der Quelltext des Moduls wird geladen.

Das Fenster der globalen Variablen <A 3>

Jedes Modul kennt seine eigenen Variablen. Wählen Sie ein Modul aus, in dem Sie im Modulfenster auf den Namen (nicht auf das Ikon) eines Moduls klicken. Wählen Sie danach aus dem Menü **Fenster** den Punkt **„Globale Variablen"** aus. Es öffnet sich ein Variablenfenster, in dem Sie die globalen Variablen des Moduls sehen. Auch hier gibt es wieder ein paar generelle Worte zu verlieren:

Der Debugger verwendet relativ häufig die Variablenfenster. Diese Fenster bestehen aus drei Listen: Die Liste der Variablennamen, der Typen der Variablen und der momantanen Werte der Variablen.



Die beiden letzten Listen können sehr lang werden - so lang, daß Sie im Fenster nicht passend dargestellt werden können. Deshalb können diese beiden Listen nicht nur vertikal sondern auch horizontal verschoben werden. Die Liste der Variablennamen zeigt auch die aus dem Modulfenster bekannten Ikons - allerdings sehen diese Ikons etwas anders aus, um sie von Modulen unterscheiden zu können. Auch diese Ikons können Sie greifen und woanders fallen lassen - zum Beispiel im Überwachungsfenster, daß Sie ebenfalls aus dem Menü öffnen können. Oder Sie greifen das Ikon und legen es im Quelltextfenster in das Empfangsfeld. Diesmal wird nicht nur das passende Modul geladen, sondern auch noch die Quelltextstelle angezeigt, an der die Variable deklariert ist. Damit können Sie z.B. die exakte Deklaration einer Variablen nachschauen (durch den Preprozessor des C++ Compilers sehen die Deklarationen im Quelltext unter Umständen anders aus, als im Variablenfenster).

Jedes Variablenfenster hat wenigstens einen Knopf über den Listen:

Den Knopf Inspiziere.

Wählen Sie eine Variable aus, indem Sie auf den Namen der Variable klicken. Wenn Sie nun den Knopf Inspiziere drücken, öffnet sich ein neues Variablenfenster, in dem Sie nun die Variable näher betrachten können. Variablen mit einfachen Typen (WORD, ULONG etc) sind nicht sinnvoll zu

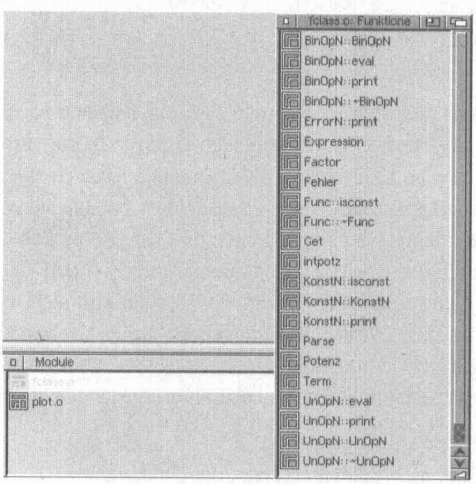
inspizieren, es wird nur der Eintrag aus dem Variablenfenster wiederholt, aber Variablen mit Strukturtypen (struct, union, class, Record) oder Pointertypen und Referenzen können näher betrachtet werden, indem von strukturierten Typen jedes Feld dargestellt wird und Pointer dereferenziert werden.

Von Arrays wird jeder Index einzeln angezeigt. Diese so dargestellten Einträge können im Inspektionsfenster ebenfalls wieder inspiziert werden, z.B. um eine ganze Liste anhand der Pointer zu verfolgen, oder auch Felder von Strukturen näher zu betrachten.

Dabei können Sie zwischen zwei Varianten der Inspektion wählen: Drücken Sie den Inspiziere Knopf mit der Shifttaste zusammen, so wird der bisher dargestellte Text im Inspektionsfenster durch den neuen Inhalt ersetzt. Drücken Sie allerdings den Knopf ohne Shifttaste, wird ein neues Fenster mit dem neuen Inhalt geöffnet. Durch Schließen dieses Fensteres können Sie dann wieder mit der Inspektion an der alten Stelle fortfahren.

Das Fenster der Funktionen <A 4>

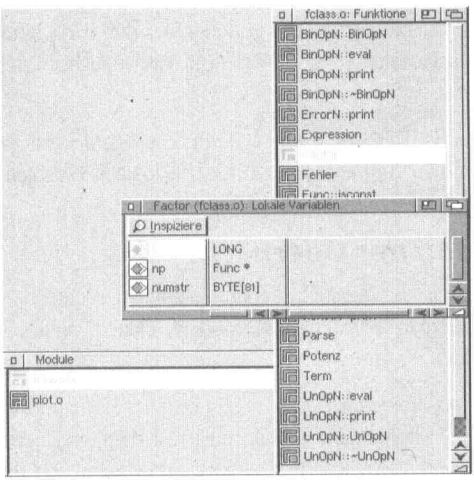
Haben Sie im Modulfenster den Namen eines Moduls ausgewählt, können Sie auch das Fenster der Funktionen dieses Moduls öffnen. Es werden alle Funktionen des Moduls angezeigt - also auch alle Methoden einer Klasse, die in dem Modul deklariert werden. Die Methoden werden dann qualifiziert ausgegeben, d.h. der Klassenname wird mit einem doppelten Doppelpunkt vor den Methodennamen geschrieben.



Die Liste der Funktionen sieht aus, wie die Liste der Module, nur mit demUnterschied, das wiederum das Ikon etwas anders gestaltet ist. Wie im Modulfenster und Variablenfenster können Sie das Ikon greifen und im Quelltextfenster (im Ikonempfangsfeld) ablegen. Wie bei Variablen wird nicht nur das Modul geladen, sondern auch der Quelltext auf den Anfang der Funktionsdeklaration positioniert.

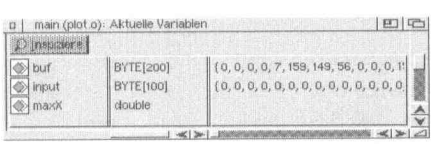
Das Fenster der lokalen Variablen <A 5>

Wählen Sie aus dem Funktionenfenster eine Funktion aus. Jetzt können Sie zu dieser Funktion ein Variablenfenster mit allen lokalen Variablen öffnen, indem Sie den entsprechenden Menüpunkt lokale Variablen aus dem Menü Fenster auswählen. Dieses Fenster ist in Form und Aufbau mit dem der globalen Variablen identisch.



Das Fenster der aktuellen Variablen <A 6>

Öffnen Sie das Fenster der aktuellen Variablen aus dem Menü „**Fenster**" mit dem Menüpunkt **„aktuelle Variablen"**. Dieses Fenster sieht genauso aus wie das Fenster der lokalen Variablen. Allerdings werden in diesem Fenster automatisch immer die Variablen der gerade aktuellen Funktion angezeigt, d.h. der Funktion, in der sich der Programmcounter gerade befindet. Dieses Fenster ist sicher neben dem Quelltextfenster das wichtigste, um sich über den momentanen Zustand des Programms zu informieren.



Das Fenster zur Überwachung <A 7>

Öffnen Sie ein Überwachungsfenster mit dem Menüpunkt **„Überwachung"** aus dem Menü **Fenster**. Sie können von diesen Fenstern beliebig viele öffnen. Dieses Fenster sieht wie die anderen Variablenfenster aus, enthält aber am Anfang keine Einträge. Sie können aber ein Ikon aus einem

beliebigen anderen Variablenfenster greifen und in die Liste der Variablennamen des Überwachungsfensters fallen lassen.

Wie im Quelltextfenster invertiert sich diese Liste, um anzuzeigen, daß diese Liste ein Ikon entgegennehmen kann. Die Variable wird dann im Überwachungsfenster genauso wie in ihrem Ausgangsfenster mit Namen, Typ und Wert dargestellt . Wenn Sie eine Variable nicht länger überwachen wollen, können Sie diese aus der Liste auswählen und mit dem Knopf ***„Entfernen"*** aus dem Fenster entfernen. Dieses Fenster ist besonders dazu geeignet, um den Bildschirm aufzuräumen und nicht unnötig viele Fenster mit globalen (und im Falle von Pascal auch lokalen) Variablen geöffnet zu haben

Die lokalen Variablen sind im Falle von C und C++ am besten im Fenster der aktuellen Variablen zu verfolgen, da man in C und C++ nur jeweils Zugriff auf die lokalen Variablen einer Funktion hat.

Die Konfiguration der Oberfläche

Alle Debugger-Fenster (Projekt-, Quelltext-, Modul-, aktuellen Variablen-, Überwachungs- und das Info-Fenster) können Sie auch beim Programmstart an einer beliebigen Position und mit Ihrer Wunschgröße wieder öffnen.

Öffnen Sie die Fenster, positionieren Sie sie und speichern Sie dann die Konfiguration aus dem Menü **„Konfiguration"**. Sie können die Fenster auch vor dem Speichern wieder schließen, dann wird zwar Position und Größe gespeichert, aber das Fenster beim Start des Debuggers nicht automatisch geöffnet.

Genauso werden auch die Position und Größe aller Fenster gespeichert, die nicht einfach so geöffnet werden können (Filerequester, Fenster der globalen Variablen, der Funktionen, der lokalen Variablen, Inspektionsfenster und Adresseneingabefenster).

Der Filerequester bekommt für alle seine Aufgaben (Laden eines Programms, Konfiguration laden und speichern) die gleiche Position und Größe. Öffnen Sie einfach den Filerequester mit dem Menüpunkt **„Konfiguration laden"**, positionieren Sie ihn und schließen ihn mit Abbruch. Beim nächsten Speichern der Konfiguration werden die Position und Größe dann mitgespeichert. Gleiches gilt auch für das Adreßeingabefenster.

Die übrigen Fenster werden per Beispiel fixiert. Öffnen Sie das erste Mal ein solches Fenster (z.B. des Fenster der globalen Variablen eines Moduls). Positionieren Sie dieses Fenster und schließen es wieder. Beim darauffolgenden Speichern der Konfiguration wird diese Position und Größe als Beispiel gespeichert. Wenn Sie also nach dem nächsten Programmstart des Debuggers ein Fenster mit globalen Variablen öffnen, wird es automatisch an der Position und in der Größe geöffnet, die Sie in ihrem Beispielfenster eingestellt hatten. Das gleiche gilt auch für die Fenster der Funktionen, der lokalen Variablen und der Inspektionsfenster.

Mit dem Menüpunkt **„Konfiguration sichern"** wird die aktuelle Konfiguration in die Datei *„PROGDIR:MSDB.prefs"* gespeichert. Sie können also den Debugger auch aus anderen Verzeichnissen oder aus der Shell starten, ohne ihre Konfiguration zu verlieren.

Mit den Menüpunkten **„Konfiguration sichern als“** und **„Konfiguration laden“** können auch mehrere verschiedene Konfigurationen benutzt werden.

Das Menü Projekt

Der erste Menüpunkt **„öffnen“** <A O> lädt und startet ein Programm. Wenn noch ein Programm läuft wird die Operation mit einer entsprechenden Fehlermeldung abgebrochen. Beenden Sie dann zuerst das laufende Programm.

Der zweite Menüpunkt **„Wieder Starten“** <A W> läßt sich nur ausführen, nachdem ein Programm beendet wurde. Dann wird dieses Programm nochmal geladen und gestartet.

Der Menüpunkt **„Über“** zeigt das Informationsfenster, mit den Copyright-Hinweisen und der Versionsnummer.

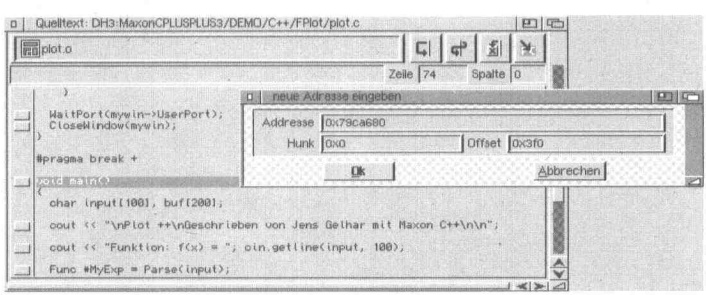
Der letzte Menüpunkt **„Debugger beenden“** <A Q> bricht ebenfalls mit einer Fehlermeldung ab, wenn noch ein Programm läuft. Ansonsten wird der Debugger beendet.

Das Menü Schritt

In diesem Menü finden Sie die Menüpunkte zur Steuerung des Programms. Die ersten vier Menüpunkte stimmen mit den Knöpfen des Quelltextfensters überein:

<A I>	einen Einzelschritt ausführen
<A R>	einen Schritt ausführen und dabei Unterprogramme schnell abarbeiten,
<A G>	bis zum nächsten Breakpoint gehen
<A E>	die aktuelle Position des Programms anzeigen.

Der nächste Menüpunkt **„Neue Adresse zeigen“** läßt sich ebenfalls im Quelltextfenster aufrufen indem Sie den Knopf zur Anzeige des PC (aktuelle Position) mit gedrückter Shifttaste anklicken.



In beiden Fällen öffnet sich das Adreßfenster, in dem Sie entweder eine Hexadezimaladresse oder eine Hunknummer mit Offset eingeben können.

Der Debugger zeigt im Quelltextfenster die passende Quelltextstelle an - vorausgesetzt er findet sie. Zum Beispiel kann damit eine Adresse eines Enforcerhits oder eines Absturzes schnell gefunden

werden, insbesondere Hunknummer und -offset, die ja unabhängig von der absoluten Adresse des Programms sind und durch Programme wie *SegTracker* ausgegeben werden, können gute Dienste in der Fehlersuche leisten.

Der letzte Menüpunkt **„Programm abbrechen"** ist etwas mit Vorsicht zu genießen und wird daher auch mit einer zusätzlichen Rückfrage quittiert. Mit diesem Menüpunkt läßt sich das Programm an jeder beliebigen Stelle abbrechen, solange es nicht gerade läuft. Hat man einen schweren Fehler gefunden, und steht nun auf einem Breakpoint, und ist sicher, daß bei der weiteren Programmausführung das Programm abstürzen wird, kann es sinnvoll sein, das Programm frühzeitig abzubrechen. Allerdings werden dadurch keine Dateien oder Fenster geschlossen, Speicher, oder irgendwelche anderen Ressourcen des Betriebssystems freigegeben.

Das Menü Fenster

Dieses Menü wurde schon in der Fensterbeschreibung erläutert. Je ein Menüpunkt öffnet eines der Fenster des Debuggers.

Das Menü Einstellungen

Mit dem Menüpunkt **„Einstellungen laden..."** kann eine neue Konfiguration der Oberfläche geladen werden.

Der Menüpunkt **„Einstellungen sichern"** speichert die aktuelle Konfiguration als Standardkonfiguration (unter dem Namen *„PROGDIR:MSDB.prefs"*).

Der Menüpunkt **„Einstellungen sichern als..."** speichert die aktuelle Konfiguration unter dem Namen, den Sie im Dateidialog eingeben.

Anhang

Der Enforcer

Der Enforcer ist ein Programm, mit dem illegale Speicherzugriffe auf Rechnern mit MMU verfolgt werden können. Viele Programmierfehler in C äußern sich auf diese Art und Weise (wegen der intensiven Benutzung der Pointer in C). Klassisch ist ein Enforcer-Hit (so wird ein erkannter illegaler Speicherzugriff genannt) in den unteren 20 oder 40 Bytes des Adreßraums, zumeist eine Folge eines Zugriffs auf den Speicher über einen nicht initialisierten Pointer, der den Wert NULL hat. Der Einsatz des Enforcers stört MaxonSDB nicht, um selbst Enforcerhits zu vermeiden, testet der Debugger jeden Speicherzugriff auf eine Variable des laufenden Programms auf den gültigen Adreßraum. Allerdings ist derzeit dieser Test noch stark eingeschränkt auf den wirklichen Speicher, der in Speicherlisten eingetragen ist, alle anderen Zugriffe werden unterdrückt. Um einen Enforcerhit im eigenen Programm zu finden, genügt es, den PC, an dem der Enforcerhit ausgelöst wurde, im Adreßfenster einzugeben, sollte diese Adresse im Programm liegen, wird der Quelltext korrekt angezeigt.

Der SegTracker

Der SegTracker arbeitet mit Enforcer zusammen, indem er die Adressen der Enforcerausgaben in die Hunknummern und -offsets des Programms umrechnet, das die Adresse enthält. Da im Adreßfenster auch Hunknummern und -offsets eingegeben werden können, und SegTracker auch die Umsetzung der Register und des Stacks beherrscht, können damit auch leicht Enforcer-Hits lokalisiert werden, die zwar durch ihr Programm verschuldet sind, aber erst durch das Betriebssystem ausgelöst werden.

inline-Funktionen

Um die beim Aufruf und bei Rückkehr einer Funktion notwendigen Maschinenbefehle einsparen zu können, wurden in C++ die inline-Funktionen eingeführt. Für diese Funktionen wird nicht nur wie bei anderen Funktionen einmal Code erzeugt, der dann bei jedem Funktionsaufruf angesprungen wird. Statt dessen merkt sich der Compiler die Zeilen der Funktion und ersetzt (fast) jeden Aufruf der Funktion durch den Funktionskörper. (Es gibt spezielle Fälle, wo diese Ersetzung nicht möglich ist, z.B. beim Aufruf der Funktion über einen Funktionspointer oder bei virtuellen Funktionen.) Analog der `register`- Anweisung in normalem C ist die Beachtung des Schlüsselwortes `inline` keine Pflicht für den Compiler, er kann das Wort schlicht ignorieren.

Durch die Verwendung von `inline` bei Funktionen wird ein Programm im allgemeinen nicht kürzer, sondern höchstens schneller. Deshalb lohnt es sich auch nur dann, Funktionen mit der Eigenschaft `inline` zu versehen, wenn diese sehr kurz sind, z.B. weil sie nur einen Wert aus einem Klassenobjekt zurückliefern. Bei einem global optimierenden Compiler allerdings kann die Verwendung des Schlüsselwortes auch ein kürzeres Programm zur Folge haben, da sich diesem neue Möglichkeiten zur Optimierung über Funktionsaufrufe hinweg bieten.

Für einen Debugger hat die Verwendung von `inline` mehrere, unangenehme Konsequenzen, die daraus resultieren, daß aus einem kurzen Stück Quelltext viele verstreute Stellen im Code werden, und somit keine eindeutige Zuordnung „Quelltext zu Code" mehr möglich ist.

Dadurch ist es nicht möglich, im Quelltext einen Haltepunkt in eine `inline`-Funktion zu setzen, und in Assembler müßten Sie einen Extra-Haltepunkt in jedes expandierte Stück setzen.

Eine `inline`-Funktion baut keinen Stack-Rahmen auf und verwendet die Register jedesmal anders, so daß ihre lokalen Variablen dem Debugger nicht zugänglich sind.

Der tiefere Sinn von `inline`-Funktionen, daß sie einem global optimierenden Compiler neue Optimierungsmöglichkeiten eröffnen, kann auch nicht genutzt werden, da ein solcher Compiler bei seiner Arbeit Quellzeilen und Variablen vertauscht, wegläßt und in ihrer Bedeutung verändert, und einen Quelltext-Debugger dabei vor kaum lösbare Probleme stellt. (Es ist sinnvoller, einen C++-Interpreter zu schreiben, als einen Debugger, mit dem man den Code im Quelltext nachvollziehen kann, den ein guter global optimierender C++-Compiler erzeugt).

MaxonSDB löst das Problem der `inline`-Funktionen folgendermaßen: Sie werden genau wie Makros behandelt. Dies bedeutet, daß der Aufruf einer `inline`-Funktion am Stück ausgeführt wird, auch wenn Sie **„Schritt/Überschreiten"** gewählt haben.

Technische Einzelheiten:

Der Debugger startet das Programm als eigenen Prozess und kommuniziert dann über Nachrichten mit ihm. Der Vorteil dieser Methode liegt darin, daß das Programm den Debugger nicht stört, und umgekehrt. Einzige Einschränkung: Dem Programm steht beim Debuggen ein Signal weniger zur Verfügung, als beim freien Lauf. Allerdings sollte kein Programm alle 16 freien Signale benötigen, oder ein Signal mit einer bestimmten Nummer belegen. Ein weiterer Vorteil ist die Möglichkeit, auch in ein laufendes Programm Breakpoints setzen, oder Variablen inspizieren, oder in jeder anderen Art mit dem Debugger weiter arbeiten zu können. Ein Nachteil soll allerdings auch nicht verschwiegen werden: DieKommunikation zwischen den Prozessen kostet ihre Zeit, und so kann die Abarbeitung eines Einzelschritts, der viele einzelne Maschinenbefehle bedeutet, eine kurze Weile dauern.

Forbid() und Disable()

Falls ein durch diese Funktionen geschützter Bereich in Einzelschritten abgearbeitet wird, wird dieser Schutz bei jedem Schritt durchbrochen, da Ihr Programm innerhalb von MaxonSDB `Wait()` aufruft (aufrufen muß!), um den Befehl zum nächsten Einzelschritt von der Benutzeroberfläche von MaxonSDB zu erhalten. Dieses Problem würde aber genauso auftreten, wenn MaxonSDB nur einen Prozeß für seine Tätigkeit verwenden würde, da der Debugger ja letztendlich auf Intuition warten muß, um das von Ihnen eingegebene Einzelschrittkommando zu erhalten.

Ein mit obigen Funktionen geklammerter Programmabschnitt, der mit **„Einzelschritt"** oder **„Überschreiten"** abgearbeitet wird, bleibt aber geschützt, falls Ihr Programm in diesem geschützten Abschnitt nicht auf einen Haltepunkt stößt. Auch innerhalb von Funktionen, die mit **„Überschreiten"** am Stück ausgeführt werden, bleibt der Schutz durch `Forbid()` oder `Disable()` erhalten.

Die Prozessoren 68020,68030 und 68040

MaxonSDB berücksichtigt automatisch den in Ihrem Amiga gerade benutzten Prozessor (so wie es jedes Programm tun sollte und leicht kann). Darüberhinaus kümmert er sich um den Cache dieser Prozessoren, wenn er z.B. Haltepunkte im Speicher installiert oder entfernt. Unter Kickstart 2.0 verwendet er die hierfür vorgesehenen Funktionen der Exec-Library, so daß es auch auf Amigas mit zusätzlichem Cache keine Probleme geben sollte (falls es solche Amigas überhaupt gibt).

Sicherheitsabfragen

Programm normal beendet

Das debuggte Programm wurde normal beendet. Der Dialog erlaubt außer dem Klick auf ***„OK"*** auch die Antwort ***„Debugger verlassen"*** woraufhin der Debugger beendet wird.

Es läuft noch ein Programm. Bitte beenden Sie dieses zuerst.

Bevor Sie ein neues Programm laden können, müssen Sie zuerst das laufende Programm beenden.

Abhilfe: Im Normalfall sollten Sie das Programm beenden, indem Sie das Programm mit **„Gehe"** (ohne gesetzte Breakpoints) schnell ablaufen lassen, bis es sich selbst beendet. Sie können das Programm auch mit dem Menüpunkt **„Programm abbrechen"** beenden

Programm wirklich abbrechen? Dadurch werden die Ressourcen...

Wenn Sie das Programm abbrechen werden die Ressourcen (Fenster, Speicher, Dateien,...) des Programmes nicht automatisch freigegeben oder geschlossen. Insbesondere bei Fenster kann es zu Systemabstürzen kommen, wenn ein freigegebenes aber immer noch dargestelltes Fenster aktiviert wird.

MaxonSDB kann noch nicht verlassen werden, da das...

Diese Meldung erscheint, wenn Sie den Debugger beenden wollen, bevor das debuggte Programm beendet wurde.

Abhilfe: Im Normalfall sollten Sie das Programm beenden, indem Sie das Programm mit **„Gehe"** (ohne gesetzte Braekpoints) schnell ablaufen lassen, bis es sich selbst beendet. Sie können das Programm auch mit dem Menüpunkt **„Programm abbrechen"** beenden

Fehlermeldungen

Es ist eine Ausnahme aufgetreten.
Das Programm muß vorzeitig beendet werden.

Der Debugger fängt jede Ausnahme (Exception) der CPU ab, wodurch das Programm beendet wird.

Abhilfe: Mit Einzelschritten den genauen Fehlerort suchen und das Programm korrigieren

Die Datei „xxx" konnte nicht geladen werden.

Diese Fehlermeldung tritt immer dann auf, wenn eine Quelltextdatei nicht geladen werden konnte. Hierfür gibt es mehrer Ursachen:

1. Die Quelltext-Datei wurde nach dem Compilieren gelöscht oder die Diskette befindet sich nicht im Laufwerk.

2. Der Debugger wurde aus einem anderen Verzeichnis als der Compiler gestartet. Die MDBI-Dateien (die unter anderem alle Quelltextnamen zu einem Modul enthalten), zeigen häufig mit relativen Pfadnamen auf die Quelltextdateien. Wurde der Debugger aus einem anderen Verzeichnis als der Compiler gestartet, ist es wahrscheinlich, daß der Debugger die Quelltextdateien nicht finden kann.

Insbesondere beim Debugger-Start aus der Shell kann es leicht passieren, daß Compiler und Debugger aus verschiedenen Verzeichnissen gestartet werden.

Abhilfe: Überprüfen Sie ob die Quelltext-Datei vorhanden ist und starten Sie den Debugger aus dem gleichen Verzeichnis aus dem Sie auch den Compiler starten.

Das Programm „xxx" konnte nicht geladen werden.

Das zu debuggende Programm konnte nicht geladen werden. Entweder steht keine ausführbare Datei mit dem angegebenen Namen zur Verfügung oder es ist keine passende MDBG-Datei vorhanden.

Abhilfe: Stellen Sie sicher, daß eine Exe-Datei und die Debug-Dateien erzeugt werden und kompilieren das Programm neu.

Das Programm „xxx" konnte nicht gestartet werden.

Das korrekt geladene Programm konnte nicht gestartet werden. Dieser Fehler kann nur durch Speicherplatzmangel auftreten.

AMIGA

MaxonC++

HotHelp 3

MAXON
computer

1. Über diese Anleitung

HotHelp liegt mittlerweile in der Version 3 vor. Falls Sie schon mit einer früheren Version des Programms vertraut sind, wird Sie die folgende Einführung und das Kapitel über die Bedienung sicherlich nicht besonders interessieren, da diese sich an den HotHelp-Einsteiger wenden. Um so interessanter dürfte für Sie jedoch das Kapitel sein, in dem die Neuerungen und Änderungen gegenüber Version 2 zusammengefaßt sind - dort sollten Sie unbedingt einen (oder auch zwei oder mehr) Blicke hineinwerfen!

Sie werden im übrigen feststellen, daß die Anleitung nicht mehr so detailliert auf HotHelp eingeht wie bei Version 2 - das ist auch nicht mehr notwendig, da HotHelp sich inzwischen komplett selbst erklärt! Das Handbuch beschreibt daher nur noch die Fähigkeiten und größeren Zusammenhänge innerhalb des HotHelp-Systems - für Detailfragen, die sich erst während der Benutzung eines Programms stellen (was genau macht der Schieberegler hier und wofür ist dieses Blättersymbol dort nützlich?) kann dann jederzeit ein Hilfstext abgerufen werden.

Sollten sich nach Drucklegung dieser Anleitung noch kurzfristig Änderungen an den Programmen ergeben haben, finden Sie auf der ersten Installations-Diskette noch eine Datei mit dem Namen "Liesmich", die Sie sich ebenfalls ansehen sollten.

2. Jederzeit hilfsbereit

Vielleicht gehören Sie ja zu den Leuten, die alles, was sie zum Arbeiten mit dem Amiga brauchen, in einem geradezu gigantischen Gedächtnis gespeichert haben. In diesem Fall haben Sie sich HotHelp leider umsonst gekauft - sorry! Verfügen Sie jedoch über kein fotografisches Gedächtnis, dann brauchen Sie HotHelp!

HotHelp ist eine Art elektronische Bibliothek, die Ihnen die gesuchten Informationen genau da zur Verfügung stellt, wo Sie sie auch benötigen - nämlich am Computer! Das Programm kann jederzeit über einen einfachen Tastendruck aktiviert werden und öffnet dann ein Fenster, in dem Sie sich die einzelnen Hilfstexte anzeigen lassen können.

Wie in einer Bibliothek gibt es verschiedene Bücher, die hier als Projekte bezeichnet werden. Ähnlich einem normalen Lexikon enthält jedes Projekt eine Reihe von Hilfstexten zu ganz unterschiedlichen Stichworten (diese werden hier als Schlüssel bezeichnet). Jeder Text kann noch eine Reihe von Querverweisen auf verwandte Themen enthalten; wird ein solcher Verweis mit der Maus oder der Tastatur ausgewählt, stellt HotHelp den dazu passenden Hilfstext dar (Hypertext-Prinzip). Entgegen einem herkömmlichen Lexikon übernimmt dabei jedoch der Amiga die umständliche Suche und Blätterei, so daß Sie sich ganz auf die Texte konzentrieren können!

HotHelp bietet dabei für jeden Amiga-Anwender etwas:

- Der Einsteiger findet im Glossar Erklärungen zu mehr als 500 verschiedenen Begriffen des typischen Amiga-Fachchinesisch (bzw. -englisch). Wenn Sie sich schon immer gefragt haben, welche Vorteile Ihnen Multitasking eigentlich bietet, was ein Guru mit dem Amiga zu tun hat, was die Aufgabe eines Betriebssystems ist, worum es sich bei einem Gadget oder einem Drawer handelt oder was bei einem Absturz genau passiert, finden Sie hier die Antworten auf Ihre Fragen.
- Für den fortgeschrittenen Amiga-Benutzer, der seinen Computer nicht nur über die Workbench steuern möchte, ist sicherlich die ausführliche Beschreibung aller Shell-Befehle und die Erklärung der Programmiersprache ARexx interessant (letztere ist schon seit einiger Zeit Bestandteil des Amiga-Betriebssystems und hat das veraltete AmigaBASIC abgelöst).
- Programmierer finden ausführliche Informationen über die Programmiersprache C (ANSI-Standard), den IFF-Standard sowie eine umfassende Dokumentation aller Betriebssystem-Funktionen, -Strukturen und -Konstanten bis einschließlich OS 3.0. Zusammen mit der Möglichkeit, Hilfe zum aktuellen Wort im gerade editierten Text abzurufen, bietet HotHelp damit eine enorme Arbeitserleichterung, da das lästige Nachschlagen in den Amiga-Handbüchern völlig entfällt!

3. Eigenschaften von HotHelp

Neben HotHelp, das jetzt in der neuen Version 3 vorliegt, gibt es noch einige andere ähnliche Programme auf dem Markt - unter anderem auch das von Commodore favorisierte AmigaGuide-System, das neuerdings auch Bestandteil des Betriebssystems ist. Hier soll nicht auf die verschiedenen Vor- oder Nachteile der anderen Produkte eingegangen werden (schließlich ist vergleichende Werbung ja auch verboten) - sie finden hier einfach eine Übersicht über alle Vorzüge von HotHelp. Vergleichen Sie diese ruhig einmal mit den Leistungsmerkmalen der Konkurrenzprodukte; Sie werden feststellen, daß HotHelp der Porsche unter den Hilfssystemen für den Amiga ist!

- HotHelp wird mit umfangreichen Hilfstexten zu einer ganzen Reihe interessanter Themen ausgeliefert (Glossar, Shell, ARexx, ANSI-C, IFF, Libs & Devs 3.0). Da die neue Version auch Dateien im AmigaGuide-Format akzeptiert, können auch die Hilfsdateien anderer Programme, die in diesem Format vorliegen, über HotHelp angezeigt werden.

- HotHelp ist auf allen Betriebssystemen ab Version 1.2 lauffähig. Alle Programme wurden anhand der von Commodore vorgeschriebenen Entwicklungsrichtlinien erstellt, mit Hilfe der Commodore-Testprogramme überprüft und haben sich als ausgesprochen stabil erwiesen.

- HotHelp-Fenster können aus jeder systemkonform programmierten Anwendung heraus aufgerufen werden und stehen somit während der Arbeit am Amiga jederzeit zur Verfügung, sobald sie benötigt werden. HotHelp unterstützt jede Bildschirmauflösung und jeden Zeichensatz fester Breite; der Fließtext paßt sich an jede Fensterbreite optimal an. Auf Wunsch kann das Programm Zeichensatz und Farben automatisch an den aktuellen Bildschirm anpassen.

- In HotHelp-Hilfstexten können Texte und Grafiken miteinander vermischt werden, wodurch die Texte wesentlich attraktiver gestaltet werden können.

- Hilfstexte können ausgedruckt, in den aktuellen Text Ihres Editors, den Zwischenspeicher (Clipboard) oder eine Datei geschrieben sowie direkt als Eingabe in ein laufendes Programm verwendet werden.

- Auf Tastendruck hin kann ein Hilfstext zum Wort unter dem Cursor in den Editoren Edward (MAXON), CygnusEd (CygnusSoft) oder TurboText (Oxxi) abgerufen werden. Diese Möglichkeit ist speziell für Programmierer sehr interessant. Außerdem kann Hilfe zum aktuellen Inhalt des Zwischenspeichers angefordert werden.

- HotHelp bietet die einzigartigen Onlineprojekte, bei denen Hilfstexte ohne den Umweg über die Übersetzung per Knopfdruck aus einem Editor heraus definiert und jederzeit wieder abgerufen werden können.

- Durch Komprimierung der Hilfstexte werden Diskettenplatz und Ladezeiten gespart.

- Die Erstellung eigener Projekte mit Hilfstexten ist nun durch zwei neue Zusatzprogramme sehr einfach geworden. Sie können dort mit Hilfe einer editor-ähnlichen grafischen Oberfläche über

Maus oder Tastatur alle erforderlichen Kennungen in den Text einfügen, die HotHelp benötigt, um z.B. Schlüsselbegriffe vom übrigen Text unterscheiden zu können. Sie finden dazu eine ausführliche Einführung in die Erstellung von Projekten, die Ihnen Schritt für Schritt erklärt, wie Sie Hilfsdateien mit eigenen Texten erzeugen können.

☞ Die Geschwindigkeit der Übersetzung selbst wurde gegenüber der letzten Version mehr als verdoppelt. Auch die Überprüfung eigener Projekte läuft jetzt mehr als doppelt so schnell ab wie bisher; das Testmodul wurde erweitert und meldet nun (neben dem Hinweis auf fehlende Schlüssel) noch eine Reihe anderer möglicher Fehlerquellen.

☞ Für die Darstellung von Quelltexten, Tabellen und ähnlichem in einem HotHelp-Projekt kann die Fließtext-Darstellung abgeschaltet werden.

☞ Das neue Ansichten-Konzept ermöglicht es, mehrere Texte gleichzeitig in einem HotHelp-Fenster zu halten und über Maus oder Tastatur zwischen den Texten hin- und herzuwechseln. Die verschiedenen Textansichten sind vollkommen unabhängig voneinander.

☞ Die ARexx-Schnittstelle erlaubt es, HotHelp von jedem ARexx-fähigen Programm aus anzusteuern.

☞ Über die neue Suchfunktion können Sie jetzt beliebige Begriffe innerhalb eines Textes suchen. Darüber hinaus haben Sie nun auch die Möglichkeit, alle Texte eines Projektes (oder auch nur einen bestimmten Teil der Texte) nach einem beliebigen Begriff zu durchsuchen, der nicht als Schlüssel innerhalb des Projektes definiert sein muß.

☞ Es können bis zu fünf Lesezeichen gesetzt werden, zu denen Sie später wieder auf Tastendruck zurückkehren können.

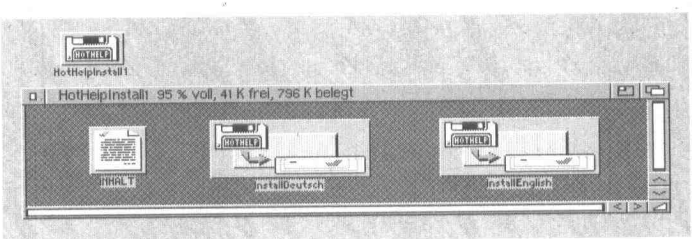
☞ Die Auswahl mehrerer Projekte und Schlüssel gleichzeitig wird durch die Verwendung von Namensmustern ermöglicht, wie sie auch in der Shell verwendet werden.

4. Anforderungen

HotHelp benötigt einen Amiga mit Kickstart 1.2 oder höher, zwei Diskettenlaufwerken und 512 KByte Speicher. Sollen alle Projekte installiert werden, sollte 1 MByte Speicher und eine Festplatte vorhanden sein - ansonsten wäre dauernder Diskettenwechsel die Folge!

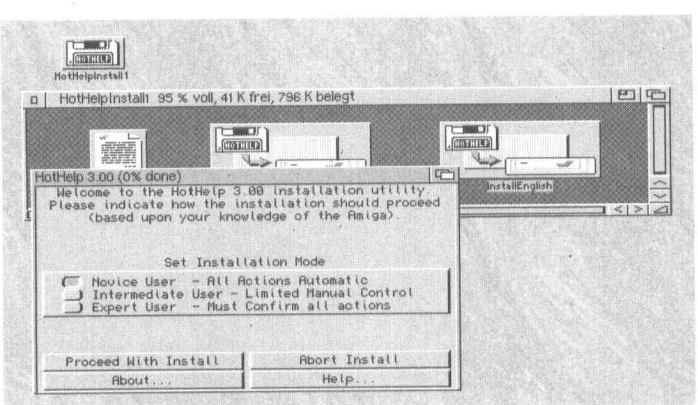
5. Installation

Damit HotHelp Ihnen jederzeit auf Tastendruck zur Verfügung steht, muß es auf Ihrer normalen Startdiskette bzw. Festplatte installiert werden. Zu diesem Zweck wird das Installationsprogramm 'Installer' von Commodore verwendet. Es ist sehr einfach zu bedienen, da es sich der Amiga-Erfahrung des Anwenders anpassen kann.



5.1. Installation auf Festplatte

Wenn Sie eine Festplatte besitzen, sind keine weiteren Vorkehrungen erforderlich. Sie sollten sich lediglich vergewissern, daß auf der Platte noch mindestens 4 MByte freier Speicherplatz vorhanden ist - soviel benötigt das komplette HotHelp-System in etwa.



5.2. Installation auf Disketten

Anwender ohne Festplatte können die volle Leistungsfähigkeit von HotHelp kaum ausnutzen - schließlich benötigen die gesamten Hilfsdateien schon fast zwei MByte an Speicher (das sind über zwei Millionen Zeichen). Allein diese Informationen belegen also schon drei Disketten! Möchten Sie dann wirklich Hilfstexte aus allen Projekten abfragen, werden Sie schnell zum Diskjockey.

Als Einsteiger (mal vorausgesetzt, daß alle Profis Festplatten besitzen...) haben Sie aber z.B. sicher kein Interesse an den Projekten, die sich lediglich mit Fragen der Programmierung beschäftigen, so daß Sie auf die Installation der entsprechenden Projekte verzichten können. Dadurch reduziert sich der Platzbedarf ganz erheblich! Sie haben während der Installation die Gelegenheit, unerwünschte Projekte auszuschließen. Nutzen Sie diese Möglichkeit!

Je nachdem, wieviel Daten Sie installieren wollen, werden bis zu vier formatierte Leerdisketten benötigt (Hinweise zum Formatieren von Disketten entnehmen Sie bitte Ihrem Amiga-Benutzerhandbuch). Stellen Sie sicher, daß alle Disketten unterschiedliche Namen tragen (z.B. "HotHelp 1", "HotHelp 2", ...) und daß keine Disk den Namen "HotHelp" trägt. Sie sollten hier keinen Fehler machen, da ansonsten später unangenehme und schwer zu behebende Probleme auftreten können.

Weiterhin müssen Sie noch sicherstellen, daß auf Ihrer Start-Diskette (das ist die Disk, die Sie üblicherweise nach dem Einschalten des Amiga einlegen, um das System zu starten) noch mindestens 130 KByte freier Speicherplatz verfügbar sind. Sie können dies erreichen, indem Sie unbenutzte Programme löschen oder selten benutzte auf andere Disketten auslagern.

Haben Sie schließlich alle Vorbereitungen abgeschlossen, starten Sie Ihren Amiga bitte mit der bereinigten Startdiskette neu.

5.3. Der Installationsvorgang

Um die Installation zu starten, legen Sie die HotHelp-Diskette 1 bitte in ein beliebiges Laufwerk und öffnen die Disk durch einen Doppelklick auf das erscheinende Piktogramm. Die Disk enthält drei weitere Piktogramme: 'INHALT', 'InstallDeutsch' und 'InstallEnglish'. Die Notwendigkeit für zwei Installationsprogramme ergibt sich aus der Tatsache, daß der Amiga erst seit kurzem die Verwendung unterschiedlicher Sprachen gestattet. Die beiden Programme unterscheiden sich daher nur in einer Hinsicht: in der englischen Version sind die Beschriftungen von Symbolen und bestimmte Standardtexte in englischer Sprache gehalten, während in der deutschen Version alle Texte in Deutsch erscheinen. Installiert wird in beiden Fällen dasselbe HotHelp!

Als Faustregel gilt: wenn Sie auf Ihrer Workbench (z.B. in der Titelzeile) deutsche Texte sehen, können Sie auch die deutsche Installation verwenden - ansonsten müssen Sie leider auf die englische zurückgreifen. Falls Sie mit dem Englischen nicht vertraut sind, finden Sie am Ende dieses Kapitels einen Abschnitt, in dem die wichtigsten Begriffe, die während der Installation auftreten können, erklärt werden.

Starten Sie nun die Installation durch einen Doppelklick auf das entsprechende Piktogramm. Nach einer einleitenden Nachricht erreichen Sie das Startfenster, in dem Sie den Modus der Installation festlegen müssen. Sie haben dabei die Möglichkeit, zwischen 'Einsteiger', 'Geübter Benutzer' und 'Experte' zu unterscheiden.

Im Einsteiger-Modus wird davon ausgegangen, daß Sie über eine Festplatte verfügen, auf die HotHelp dann installiert wird. Sie müssen dazu anschließend noch den Namen dieser Platte eingeben. Es handelt sich dabei um den Namen, der auch auf der Workbench unter dem Piktogramm Ihrer Festplatte sichtbar ist, gefolgt von einem Doppelpunkt. Alle weiteren Aktionen werden dann automatisch durchgeführt - Sie werden lediglich noch aufgefordert, entsprechende Disketten einzulegen.

Im Modus für geübte Benutzer muß eine Reihe von Abfragen beantwortet werden - welche Funktionsbibliotheken installiert werden sollen, in welchem Verzeichnis die Daten für HotHelp abgelegt werden müssen und ähnliches. Normalerweise können Sie die vom Programm gemachten Vorgaben einfach akzeptieren. Sie sollten diesen Modus verwenden, wenn Sie mehr Kontrolle über den Verlauf der Installation wünschen - dieser Modus ermöglicht es z.B. auch, HotHelp auf Disketten zu installieren.

Im Experten-Modus schließlich muß jede einzelne Aktion bestätigt werden. Sie haben dadurch die volle Kontrolle über den Installationsvorgang, werden jedoch auch mit einer Reihe eigentlich unwichtiger Details belästigt.

Falls Sie die Modi für geübte Benutzer oder Experten verwenden, können Sie in den meisten Fenstern über das Hilfe-Symbol einen ausführlichen Hilfstext über den aktuellen Status und den Sinn des entsprechenden Fensters abfragen. Scheuen Sie sich nicht (auch wenn Sie als Experte installieren), dieses Symbol zu betätigen, wenn Ihnen die Bedeutung der aktuellen Abfrage für die Installation nicht ganz klar sein sollte.

Unabhängig vom eingestellten Modus geht das Installationsprogramm möglichst genau auf Ihre aktuelle Konfiguration ein. Insbesondere wird auch erkannt, wenn Sie HotHelp schon einmal installiert haben. Das Skript sorgt dann dafür, daß die alte HotHelp-Version problemlos durch die neue ersetzt wird.

Wenn Sie die Installation erfolgreich beendet haben, müssen Sie Ihren Rechner neu starten (Ctrl-Amiga-Amiga), um HotHelp zu aktivieren. Das Programm steht Ihnen von nun an nach jedem Neustart automatisch zur Verfügung. Sie erkennen dies daran, daß sich während des Starts kurzzeitig ein Fenster mit dem Text 'HotKeyHelp wird installiert' öffnet.

5.4. Installation von Hand

Manche Experten trauen ja grundsätzlich keinem Installations-Skript und wollen immer alles von Hand (bzw. von Shell) kopieren. Davon sei Ihnen jedoch an dieser Stelle entschieden abgeraten! Das Installations-Skript ist nicht umsonst recht lang geworden - viele Elemente von HotHelp sind auf den Installationsdisks noch komprimiert, die Projekte müssen noch nachbearbeitet werden, es muß eine Voreinstellungsdatei erzeugt, eine Änderung muß in die Datei 'User-Startup' eingetragen werden und noch einiges mehr.

Hier nur noch ein paar Hinweise auf die Aktionen während der Installation:

Ins Verzeichnis LIBS: werden zwei oder drei Libraries kopiert. Das Verzeichnis L: nimmt zwei Binärdateien, die sogenannten HotHelp-Handler, auf. Das ausführbare Programm HotKeyHelp wird ins C:-Verzeichnis kopiert. Wenn Ihr Betriebssystem schon verschiedene Sprachen unterstützt, werden zwei Katalogdateien in das Verzeichnis LOCALE:Catalogs/Deutsch übertragen. Die ausgewählten HotHelpTools, die Beispiel-Projekte und -Programme werden in den dafür ausgewählten Verzeichnissen abgelegt; alle übrigen Dateien werden in dem für die HotHelp-Daten vorgesehenen Verzeichnis untergebracht. Schließlich werden noch einige Befehle in die Datei 'User-Startup' eingebaut. Dabei wird das logische Gerät HOTHELP: dem Verzeichnis mit den HotHelp-Daten zugeordnet.

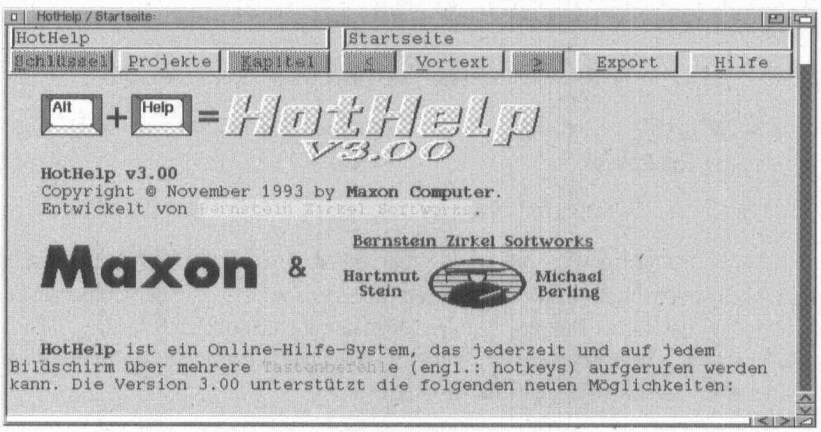
5.5. Installation mit englischer Benutzerführung

Falls Sie noch mit Betriebssystem-Versionen vor 2.1 arbeiten, unterstützt Ihr Amiga lediglich die englische Sprache. In diesem Fall müssen Sie leider das Installationsskript 'InstallEnglish' benutzen, in dem alle Symbole in Englisch beschriftet sind. Hier soll dazu kurz auf die wichtigsten Begriffe eingegangen werden.

- Proceed, Proceed With Install: Fortsetzen der Installation. Falls im aktuellen Fenster Einstellungen vorgenommen werden konnten, werden diese übernommen und der nächste Schritt der Installation eingeleitet.
- Abort, Abort Install: Abbruch der Installation. Hiermit wird der gesamte Installationsvorgang vorzeitig beendet.
- Help: Zeigt einen (deutschen) Text mit Informationen über das aktuelle Fenster an.
- Novice User, Intermediate User, Expert User: Dies sind die drei Installationsmodi Einsteiger, geübter Benutzer und Experte.
- Yes, No: Ja bzw. Nein bei einer einfachen Abfrage.
- Selected Drawer, Parent Drawer, Make new Drawer, Show Drives: Diese Begriffe tauchen bei der Auswahl einer Schublade (Drawer) auf. 'Selected Drawer' ist der Name der momentan eingestellten Schublade; 'Parent Drawer' zeigt den Inhalt der übergeordneten Schublade (Mutterverzeichnis) an, über 'Make new Drawer' können Sie eine neue Schublade erzeugen und 'Show Drives' zeigt eine Übersicht aller angeschlossenen Geräte an.

6. Die Bedienung von HotHelp

Nachdem die Installation abgeschlossen wurde und Sie einen Neustart ausgeführt haben, brennen Sie sicher schon darauf, das erste HotHelp-Fenster zu sehen. Drücken Sie dazu eine der beiden Alt-Tasten (am unteren Rand der Tastatur neben den Tasten mit den Amiga-Symbolen). Halten Sie die Taste gedrückt und drücken Sie noch zusätzlich die Help-Taste (oberhalb der vier mit Pfeilen beschrifteten Cursortasten). Schon sehen Sie Ihr erstes HotHelp-Fenster vor sich auf dem Bildschirm auftauchen.



Eine solche Tastenkombination wird als Hotkey (heiße Tasse ...äh... Taste) bezeichnet, da sie augenblicklich eine Reaktion hervorruft. Sie können diesen Hotkey jederzeit verwenden, um ein Hilfsfenster zu öffnen. Das Fenster erscheint immer auf der Arbeitsfläche des Programms, mit dem Sie gerade arbeiten.

ACHTUNG:

Einige Programme verwenden ihre Arbeitsfläche, ohne damit zu rechnen, daß diese auch von anderen Programmen benutzt wird. Aus diesem Grund kann es vorkommen, daß der Bildschirmaufbau des Programms mehr oder weniger stark durcheinander kommt (Datenverluste durch Abstürze sollten dadurch allerdings nicht auftreten).

Bei neueren Programmen können Sie das Problem beheben, indem Sie das Programm anweisen, seinen Bildschirm der Öffentlichkeit zur Verfügung zu stellen. Dafür müßte eine Einstellmöglichkeit wie z.B. 'Bildschirm öffentlich machen' oder 'Make screen public' vorhanden sein. Sollte das von Ihnen verwendete Programm keine solche Möglichkeit bieten, sollten Sie HotHelp nur noch vorsichtig auf dem Bildschirm dieses Programms einsetzen!

ACHTUNG:

Schließen Sie niemals einen Bildschirm, auf dem sich noch ein geöffnetes HotHelp-Fenster befindet, da Sie HotHelp damit den Boden unter den Füßen wegziehen! Dies hat meist einen bodenlosen Absturz zur Folge...

6.1. Die Titelzeile des HotHelp-Fensters

Das neu geöffnete Fenster zeigt Ihnen einen einleitenden Text über HotHelp selbst an. In der Titelzeile des Fensters sehen Sie den Titel dieses Textes - es handelt sich um die Startseite aus dem HotHelp-Projekt. Um wieder den Bibliotheksvergleich zu bemühen: Sie sehen jetzt die Einleitung des Buches, das sich speziell mit den Fähigkeiten von HotHelp befaßt. Wir können also festhalten: in der Titelzeile wird immer angezeigt, welchen Text Sie gerade lesen.

6.2. Rollsymbole

Im rechten und unteren Rand des Fensters sehen Sie zwei Rollsymbole vor sich, die dazu dienen, den sichtbaren Textausschnitt einzustellen. Das Symbol im unteren Fensterrahmen wird dabei nur benötigt, wenn Sie das Fenster so weit verkleinern, daß ein Text nicht mehr in eine sichtbare Zeile paßt. Da HotHelp bei der Ausgabe üblicherweise mit Fließtext arbeitet (dieser wird bei jeder Änderung der Fenstergröße so gut wie möglich an die neue Größe angepaßt), passiert dies nur relativ selten. Anders verhält es sich mit Grafiken - da diese natürlich nicht umgebrochen werden können, kommt hier das Rollsymbol im unteren Fensterrand häufiger zum Einsatz. Sie können das überprüfen, indem Sie jetzt das Fenster mit der HotHelp-Startseite auf halbe Breite bringen.

Wesentlich wichtiger ist das Rollsymbol im rechten Rand - es wird immer benötigt, wenn der Text mehr Zeilen beansprucht als im Fenster Platz sind (was bei den meisten Texten der Fall ist, wie auch bei der HotHelp-Startseite). Sie können sich dann mit Hilfe dieses Symbols den ganzen Text Zeile für Zeile anzeigen lassen.

Sie können den Text auch direkt über die Maus verschieben. Klicken Sie dazu einfach das Fenster ober- oder unterhalb des Textbereiches an: der sichtbare Textausschnitt verschiebt sich um eine Zeile. Halten Sie die Maustaste gedrückt, so wird der Rollvorgang nach kurzer Zeit automatisch wiederholt. Er bricht ab, sobald Sie die Taste wieder loslassen.

Eine weitere Möglichkeit besteht darin, eine beliebige Position im Textbereich anzuklicken und die Maustaste gedrückt zu halten. Ziehen Sie die Maus nun aus dem Textbereich heraus, wird der Text in die entsprechende Richtung verschoben.

6.3. Querverweise

Querverweise auf andere Texte bieten Ihnen eine einfache Möglichkeit, weiterführende Informationen abzurufen. Sie können an jeder Position im Text erscheinen und werden durch eine andere Farbe gekennzeichnet (normalerweise Blau ab Betriebssystemversion 2.0, sonst Orange). Einer der Querverweise wird immer invertiert dargestellt (also nicht Blau auf Grau, sondern umgekehrt) - dies ist der aktuelle Querverweis. Sie können diesen mit Hilfe der Maus oder der Tastatur setzen und den zugehörigen Text abfragen. Ein Querverweis kann sich auf einen Text im aktuellen oder einem anderen Projekt beziehen.

Bei Verwendung der Maus genügt es, einen Querverweis anzuklicken, um ihn zum aktuellen Querverweis zu machen. Wenn Sie irgendwo ins Fenster klicken, dann die linke Maustaste gedrückt halten und die Maus bewegen, können Sie jeden der sichtbaren Querverweise in beliebiger Reihenfolge anwählen. Verlassen Sie dabei mit dem Mauszeiger den Textbereich, so wird der Text in die entsprechende Richtung verschoben. Sie können also sehr einfach jeden Querverweis im Text ansteuern. Durch einen Doppelklick auf einen Querverweis wird der dazu passende Text von HotHelp nachgeladen und dargestellt.

Wollen Sie die Tastatur verwenden, wählen Sie bitte zuerst den gewünschten Querverweis aus. Sie können dazu die Tabulatortaste oder die vier Cursortasten benutzen. Gegebenenfalls verschiebt

HotHelp den sichtbaren Textausschnitt, um den aktuellen Querverweis anzuzeigen. Haben Sie schließlich den gewünschten Querverweis angewählt, können Sie sich durch Betätigen der Eingabetaste (Return oder Enter) den zugehörigen Text anzeigen lassen.

Es gibt noch eine weitere Methode, einen Querverweis über die Tastatur auszuwählen. Halten Sie dazu eine der beiden Alt-Tasten gedrückt und geben dann die Anfangsbuchstaben des Querverweises ein, so wird dieser von HotHelp zum aktuellen Querverweis gemacht, dessen Hilfstext Sie dann wie üblich mit der Eingabetaste abfragen können. Dies ist recht praktisch, wenn ein langer Text viele Querverweise enthält und Sie einen dieser Querverweise auswählen möchten, aber nicht wissen, wo er im Text eigentlich auftritt - überlassen Sie einfach HotHelp die Suche!

6.4. Das Vortext-Symbol

Im Gegensatz zu einem normalen Buch merkt sich HotHelp die Seiten, die Sie bisher gelesen haben. Über das Vortext-Symbol können Sie nun ohne weiteres wieder zu den entsprechenden Texten zurückkehren. Normalerweise merkt sich das Programm bis zu 20 Schritte; sie können diese Anzahl jedoch bis auf 500 erhöhen...

6.5. Blättern im Projekt

Wie in einem richtigen Buch können Sie auch in einem HotHelp-Projekt nach Herzenslust blättern. So, wie in einem Buch Seite auf Seite folgt, folgt in HotHelps Projekten Hilfstext auf Hilfstext. Sie können die beiden mit Pfeilen beschrifteten Symbole links und rechts vom Vortext-Symbol dazu benutzen, um die Texte des Projektes der Reihe nach anzusehen. Im Gegensatz zu einem Lexikon, wo alle Texte lediglich alphabetisch sortiert sind, finden Sie hier jedoch beim Blättern einen thematischen Zusammenhang vor. Sie können sich damit durch das Blättern in einem Projekt leicht einen Überblick über ein oder mehrere Themen verschaffen. Das Vortext-Symbol bringt Sie dabei immer wieder zu den zuletzt gelesenen Texten zurück.

6.6. Schlüssel, Projekte und Muster

Am oberen Rand des Fensters befinden sich zwei Eingabefelder. Das linke der beiden ist für den Projektnamen, das rechte für den Schlüsselbegriff zuständig. Wird ein neuer Hilfstext dargestellt, trägt HotHelp die entsprechenden beiden Begriffe in diese Eingabefelder ein.

Interessant werden die beiden Felder jedoch erst, wenn Sie selbst einen Text dort eingeben. Suchen Sie z.B. nach einem bestimmten Begriff, können Sie diesen in das rechte Eingabefeld eintragen. Bestätigen Sie dann die Eingabetaste, sucht HotHelp den zu dem Schlüssel passenden Text und stellt ihn dar. Sie haben so durch die direkte Eingabe die Möglichkeit, jeden gesuchten Begriff direkt anzugeben, ohne auf einen entsprechenden Querverweis angewiesen zu sein. Falls der Begriff in einem anderen als dem eingestellten Projekt vorhanden ist, sollten Sie den Namen des entsprechenden Projektes in das linke der beiden Felder eintragen.

Möglicherweise haben Sie manchmal das Problem, daß Ihnen der genaue Wortlaut des Suchbegriffs nicht bekannt ist. Nehmen wir einmal an, Sie wollen einen Hilfstext über das Stichwort 'Online-Pro-

jekte' abrufen. Leider ist Ihnen aber die genaue Schreibweise des Begriffs unbekannt - mögliche Kandidaten wären z.B. 'Online-Projekte', 'Onlineprojekt' oder auch 'Online Projekt'. Natürlich könnten Sie jetzt alle diese Begriffe ausprobieren (obwohl erfahrungsgemäß der einzig richtige Begriff der einzige ist, der Ihnen nicht einfallen wird). Das ist jedoch eine stupide, langweilige und frustrierende Aufgabe - also wie geschaffen dafür, von einem Computer übernommen zu werden!

HotHelp unterstützt zu diesem Zweck auch Namensmuster in den Eingabefeldern. Shell-Anwendern ist dieser Begriff sicherlich vertraut: über ein Muster kann bei den Shell-Befehlen eine ganze Reihe von Dateien gleichzeitig angesprochen werden. Der Trick liegt darin, daß in den Namen verschiedene Sonderzeichen eingesetzt werden, die ganze Gruppen von Zeichen oder Zeichenfolgen abdecken. Eines dieser Sonderzeichen ist der Stern (*) - er steht für einen beliebigen Text.

Geben Sie im obigen Beispiel daher einfach '*Online*Projekt*' als Schlüsselbegriff ein. HotHelp sucht dann alle Texte heraus, deren Schlüssel die Worte 'Online' und 'Projekt' (in dieser Reihenfolge) enthalten; zwischen den Worten dürfen sich dabei beliebige Zeichenfolgen befinden. Da diese Zeichenfolgen auch leer sein dürfen, finden Sie damit garantiert alle der oben angegebenen Schlüsselvariationen (der korrekte Begriff lautet in diesem Falle übrigens 'Onlineprojekt').

Dasselbe gilt auch für das Eingabefeld für den Projektnamen. Sind Sie sich nicht sicher, in welchem Projekt die gesuchte Information sich befindet, geben Sie hier einfach einen einzelnen Stern ein. HotHelp untersucht dann alle Projekte (denn jedes Projekt hat einen Namen aus einer beliebigen Zeichenfolge und stimmt somit mit dem angegebenen Muster überein) und zeigt alle gefundenen Schlüssel.

Natürlich kann es auch passieren, daß mehr als ein Schlüssel zu dem eingegebenen Muster paßt. In diesem Fall stellt HotHelp zuerst eine nach Projekten gegliederte Übersicht über alle Schlüssel auf, aus der Sie dann den entsprechenden Querverweis wie gewöhnlich auswählen können. Einen besonderen Komfort bietet Ihnen hier dann noch das Schlüssel-Menü, das Sie mit der rechten Maustaste erreichen. Über die beiden Einträge dieses Menüs können Sie der Reihe nach alle Texte durchblättern, die zu dem angegebenen Muster passen. Über das dann aktive Schlüssel-Aktionssymbol am linken Rand der Symbolleiste können Sie die Schlüsselübersicht jederzeit wieder abrufen.

Unter Betriebssystem-Versionen vor 2.0 steht Ihnen als Musterzeichen nur der einzelne Stern zur Verfügung. In höheren Versionen können Sie alle Shell-Namensmuster verwenden - schauen Sie dazu ggf. in Ihrem Amiga-Benutzerhandbuch nach, in dem Sie eine Beschreibung der Musterzeichen finden. Oder noch einfacher: sparen Sie sich diese mühsame Suche und rufen Sie lieber den Hilfstext im Projekt 'Glossar' unter dem Schlüsselbegriff 'Namensmuster' auf! Sie sind jetzt stolzer Besitzer von HotHelp - was sollen Sie noch mit Handbüchern...

6.7. Abrufen der Projektübersicht

Über das Projekte-Aktionssymbol können Sie eine Übersicht über alle geladenen Projekte abrufen. Sie sehen dann im Fenster eine Liste aller Projektdateien, aus der Sie eine wie üblich als Querverweis auswählen können. HotHelp stellt dann den Einleitungstext des entsprechenden Projektes dar.

6.8. Die Kapitelübersicht

HotHelp-Projekte sind üblicherweise nach Themengebieten gegliedert und in entsprechende Kapitel eingeteilt (wie es sich für ein gutes Buch gehört). Über das Kapitel-Aktionssymbol können Sie nun die zum aktuellen Text passende Kapitelübersicht anfordern.

Dieses Symbol ist nicht anwählbar, wenn Sie sich bereits auf der obersten Kapitelebene (der Projekt-Einleitung oder -Startseite) befinden.

6.9. Der Export von Hilfstexten

HotHelp wacht nicht eifersüchtig wie eine Glucke über ihre Eier (bzw. Hilfstexte), sondern ist durchaus bereit, diese mit anderen Programmen zu teilen. Sie haben sogar eine ganze Reihe von Möglichkeiten, um Texte an andere Programme zu übertragen. Als erstes müssen Sie dazu jedoch über das Export-Aktionssymbol den sogenannten Exportmodus von HotHelp aktivieren. In diesem Modus sind weder die beiden Eingabefelder noch irgendwelche Querverweise verwendbar, so daß Sie den angezeigten Text nicht ändern können. Zu diesem Zweck müssen Sie den Modus erst wieder verlassen; dazu steht das Zurück-Symbol zur Verfügung.

Im nächsten Schritt wählen Sie nun bitte den gewünschten Ausschnitt des Textes aus - es ist ja durchaus möglich, daß Sie nur an einem kleinen Teil eines langen Textes interessiert sind. Dazu gibt es drei Möglichkeiten:

- Ziehen: Klicken Sie dazu mit der Maus auf den Anfang oder das Ende des gewünschten Bereiches. Halten Sie die linke Maustaste gedrückt und fahren bis zum anderen Ende des Bereiches. Falls dieses nicht sichtbar ist, können Sie den sichtbaren Textausschnitt wie üblich durch Überschreiten des Fensterrandes in die entsprechende Richtung verschieben. Lassen Sie die Maustaste dann über der gewünschten Zielzeile los.

- Erweiterte Auswahl: Klicken Sie die Anfangs- oder End-Zeile des gewünschten Bereiches an. Lassen Sie dann die linke Maustaste los, drücken eine der beiden Umschalttasten und klicken auf das andere Ende des Bereiches. Der Text zwischen den beiden angewählten Zeilen wird markiert. Da Sie zwischen den beiden Klicks auch die Möglichkeit haben, den sichtbaren Textausschnitt über die Rollsymbole zu verschieben, können Sie so sehr schnell große Blöcke auswählen.

- Doppelklick: Wenn Sie mit der Maus einen Doppelklick auf eine beliebige Zeile vornehmen, wird der gesamte aktuelle Hilfstext markiert. Auf diese Weise können Sie sich vor allem bei größeren Texten das Auf- und Abrollen sparen, das zur Markierung des Textes mit den beiden anderen Methoden notwendig ist.

Der markierte Text kann nun über die vier Aktionssymbole in der linken Hälfte der Symbolleiste exportiert werden.

Das Drucker-Symbol gibt den Text (wer hätte es gedacht) auf einen angeschlossenen Drucker aus. Der Ausdruck erfolgt anhand der Systemvoreinstellungen, die über das entsprechende Systemprogramm (Preferences bzw. Printer) gesteuert werden können.

Über das Datei-Symbol können Sie den markierten Bereich in einer normalen Textdatei abspeichern, die Sie hinterher mit jedem Editor und jeder Textverarbeitung wieder einlesen können. HotHelp öffnet dazu ein Datei-Auswahlfenster, in dem Sie den Namen der Zieldatei eingeben können.

In den Amiga-Zwischenspeicher, das sog. Clipboard, wird der Text über das dritte Symbol kopiert. Das Clipboard wird mittlerweile von vielen Programmen unterstützt, so daß Sie sich den Umweg über eine Datei sparen können.

Das vierte Symbol mit dem schlichten Titel 'Einfügen' schließt das HotHelp-Fenster automatisch. Anschließend wird dann der angewählte Text so behandelt, als hätten Sie ihn selbst über die Tastatur eingegeben. War z.B. vor dem Öffnen des HotHelp-Fensters ein Shell-Fenster aktiv, wird der markierte Text in dieses Fenster geschrieben und die Shell versucht, den Text als Shell-Befehle zu interpretieren - je nach Inhalt mit mehr oder weniger Erfolg. Gehört das zuletzt aktive Fenster hingegen zu einem Textverarbeitungsprogramm, wird der markierte Abschnitt dort in den Text eingetragen. Um den Einfügevorgang abzubrechen, genügt es, eine Maustaste zu drücken.

6.10. Verschiedene Ansichten über HotHelp

Hier soll nicht etwa die Meinung mehrerer zufriedener HotHelp-Anwender vorgestellt werden - statt dessen wird hier gezeigt, wie Sie über HotHelp Ansichten verschiedener Hilfstexte in einem Fenster zusammenfassen können. Zu diesem Zweck existiert das Ansichten-Menü.

Ansichten bieten die Möglichkeit, in einem einzigen HotHelp-Fenster beliebig viele verschiedene Texte gleichzeitig bereitzuhalten - jeder Text befindet sich in seiner eigenen Textansicht. Jede Ansicht verhält sich im Prinzip wie ein vollkommen unabhängiges HotHelp-Fenster. Die Anzahl von Ansichten, die in einem Fenster eingerichtet werden können, wird lediglich durch die aktuelle Größe des Fensters beschränkt; je größer das Fenster ist, desto mehr Ansichten finden auch darin Platz. Über die Maus oder die Tastatur kann einfach zwischen den verschiedenen Ansichten hin- und hergeschaltet werden.

Sie erkennen ein HotHelp-Fenster mit zwei oder mehr Ansichten an den Querbalken im Textbereich des HotHelp-Fensters. Diese sind jeweils mit dem Titel des Textes beschriftet, der sich in der entsprechenden Ansicht befindet. Sie können so auf einen Blick erkennen, welche Texte in den verschiedenen Ansichten geladen sind.

Bis auf eine sind alle diese Ansicht-Titelleisten mit der normalen Hintergrundfarbe gefüllt. Die einzige Ausnahme stellt hierbei die sogenannte aktive Ansicht dar - das ist diejenige, deren Inhalt gerade angezeigt wird. Ihr Titel hat einen farbigen Hintergrund. Alle Aktionen, die Sie durchführen, beziehen sich immer nur auf diese aktive Ansicht; alle anderen Ansichten werden davon nicht beeinflußt. Sie können also mit der aktiven Ansicht genauso arbeiten wie mit einem eigenen HotHelp-Fenster.

Klicken Sie nun mit der Maus auf den Titel einer anderen Ansicht, so wird diese aktiviert, und ihr Inhalt wird im Fenster dargestellt. Die vorher aktive Ansicht schrumpft bis auf ihre Titelleiste zusammen. Sie können sich auch über zwei Menübefehle vorwärts und rückwärts durch die Ansichten bewegen. Ein weiterer Menübefehl schließt und entfernt die aktuelle Ansicht wieder, wobei automatisch ihre Vorgängerin aktiviert wird.

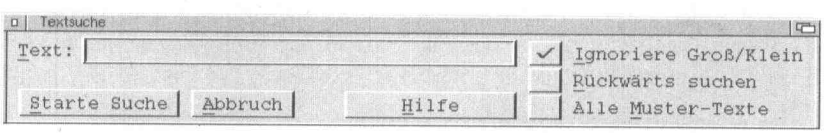
Sehr nützlich ist auch noch die Möglichkeit, eine neue Ansicht zu öffnen, in der automatisch der zum aktuellen Querverweis gehörende Hilfstext dargestellt wird. Zu diesem Zweck kann der unterste Menüpunkt 'Querverweis-Ansicht' verwendet werden.

6.11. Die integrierte Suchfunktion

Das neue Suchen-Menü bietet Ihnen die Möglichkeit, HotHelps Hilfstexte nach beliebigen Worten zu durchsuchen - ganz so, wie Sie es von Ihrem Editor oder Ihrer Textverarbeitung auch gewöhnt sind. Dies ist vor allem bei längeren Hilfstexten recht nützlich.

Die ersten drei Einträge des Suchen-Menüs finden sich so oder ähnlich in den meisten Programmen wieder: 'Suche Text...' öffnet ein Kommunikationsfenster, in dem Sie den Suchtext eingeben und verschiedene Einstellungen vornehmen können; über die beiden nächsten Menüpunkte können Sie dann das vorige oder nächste Auftreten des Suchwortes anzeigen lassen. Das gefundene Wort wird farblich invertiert dargestellt und (um es vom aktuellen Querverweis zu unterscheiden) zusätzlich mit einem Rahmen umgeben. Falls das Wort nicht gefunden wird, läßt HotHelp den Bildschirm kurz aufblitzen.

Der Menüpunkt 'Schlüsselposition' sucht nach dem ersten Auftreten des aktuellen Schlüsselbegriffes im Text und kann bei längeren Texten recht nützlich sein. Der letzte Menüpunkt 'Lösche Markierung' schließlich entfernt die Markierung des zuletzt gefundenen Wortes wieder.



Im Fenster für die Einstellung der Suchoptionen finden Sie ein Texteingabefeld, in das Sie den Suchtext eingeben können. Starten Sie die Suche dann bitte, indem Sie die Eingabe mit der Eingabetaste bestätigen oder das entsprechende Symbol auswählen. Auch die beiden oberen Auswahlfelder tauchen so oder ähnlich in jeder Textverarbeitung auf: Das obere bestimmt, ob bei der Suche die Groß-/Kleinschreibung beachtet werden soll; das darunter liegende steuert die Suchrichtung - Vorwärtssuche in Richtung des Textendes oder Rückwärtssuche auf den Anfang des Textes zu.

Am interessantesten ist jedoch das unterste Auswahlfeld mit dem Titel 'Alle Muster-Texte'. Wird dieses Symbol angewählt, beschränkt sich HotHelp bei der Suche nicht nur auf den aktuellen Text, sondern untersucht der Reihe nach alle Texte, die zum aktuellen Projekt- und Schlüssel-Muster passen. Sie haben damit die Möglichkeit, auch Begriffe, die nicht als Schlüssel definiert wurden, in HotHelp-Texten aufzufinden.

Wollen Sie z.B. alle Texte des Shell-Projektes sehen, in denen das Wort 'Brabbelt' auftaucht, geben Sie als Projektname 'Shell' und als Schlüsselmuster '*' ein. Öffnen Sie dann das Hilfsfenster, geben 'Brabbelt' als Suchtext ein und aktivieren die Suche in allen Texten. HotHelp lädt dann nacheinander alle Texte des Shell-Projektes und durchsucht sie. Natürlich ist diese Art der Suche nicht besonders schnell - dafür wissen Sie anschließend aber auch, welcher der Shell-Befehle zum Brabbeln neigt...

6.12. Verwendung von Lesezeichen

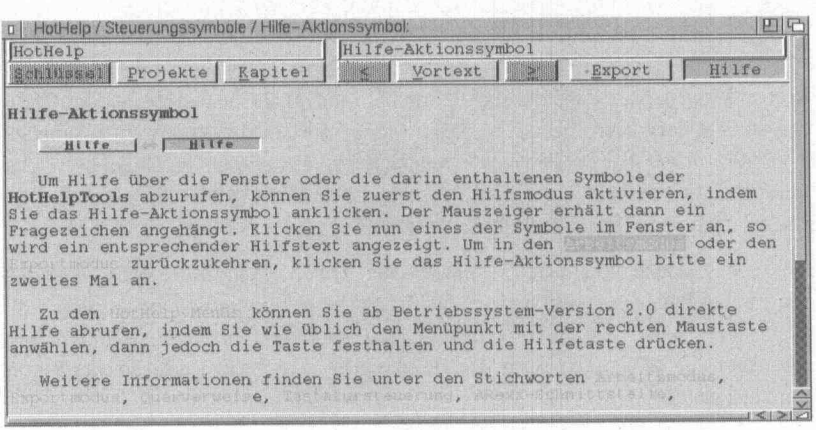
Über das Lesezeichen-Menü haben Sie die Möglichkeit, HotHelp-Texte durch elektronische Eselsohren zu kennzeichnen. Sie können bis zu fünf Marken in beliebige Texte setzen, die Sie dann jederzeit wieder abrufen können (natürlich können Sie nur Marken abrufen, die Sie vorher auch gesetzt haben).

Dies kann recht nützlich sein, wenn Sie mehrmals zwischen einer Reihe von Texten hin- und herspringen müssen. Setzen Sie in jeden der Texte eine Marke, können Sie die Texte ganz einfach wieder abrufen.

Der unterste Menüpunkt dient zum Löschen aller gesetzten Lesezeichen - er bügelt die Eselsohren wieder glatt...

6.13. Hilfe über Hilfe

Bis hierher wurde versucht, Ihnen einen möglichst umfassenden Überblick über die Bedienung von HotHelp-Fenstern zu geben. Falls Ihnen bei dieser "Tour de Force" durch HotHelp jedoch die Bedeutung des einen oder anderen Bedienungselementes wieder entfallen sein sollte, können Sie jederzeit Hilfe dazu abrufen. Zu diesem Zweck steht das Hilfe-Symbol zur Verfügung. Wenn Sie es anwählen, verwandelt der Mauszeiger sich in einen kurzen Pfeil mit einem angehängten Fragezeichen. Klicken Sie nun eines der Symbole auf dem Bildschirm an, zeigt HotHelp einen entsprechenden Hilfstext an, der die genaue Bedeutung dieses Symbols beschreibt.



Wenn Sie über Betriebssystem-Version 2.0 oder höher verfügen, können Sie auch zu den Menüs ganz einfach Hilfe abrufen: Wählen Sie dazu den Menüpunkt wie üblich mit der rechten Maustaste an. Halten Sie dann die Maustaste jedoch weiterhin fest und drücken noch zusätzlich die Hilfetaste auf der Tastatur - HotHelp zeigt Ihnen dann einen Text zu dem entsprechenden Menüpunkt an. Befand der Mauszeiger sich über dem Titel eines Menüs, wird statt dessen ein allgemeiner Text über das Menü selbst angezeigt, ohne auf einen speziellen Menüpunkt einzugehen.

Falls Sie noch nicht mit OS 2.0 arbeiten (warum eigentlich nicht?), sollten Sie sich den Text etwas genauer ansehen, der beim Betätigen des Hilfe-Symbols angezeigt wird: er enthält eine Reihe von Querverweisen, unter anderem auch auf die HotHelp-Menüs. Sie können die Texte über die Menüs dann wie gewohnt über die Querverweise abrufen und durchlesen.

Auch wenn Sie OS 2.0 oder eine höhere Version verwenden, sollten Sie sich diesen Text ruhig genauer ansehen - er enthält noch eine Reihe von Verweisen auf andere, interessante Themen, auf die in dieser Anleitung nicht weiter eingegangen wird.

Der Hilfsmodus bleibt solange aktiv, bis Sie das Hilfe-Symbol nochmals anwählen (oder bis Sie das Fenster wieder schließen).

6.14. Tastaturbedienung

HotHelp kann fast hundertprozentig über die Tastatur gesteuert werden - der Griff zur Maus ist nur in Ausnahmefällen notwendig!

Die Menüs können wie üblich durch Kombination des angezeigten Buchstabens mit der rechten Amiga-Taste angewählt werden (bzw. durch eine Zifferntaste zusammen mit der Controltaste beim Setzen der Lesezeichen).

Für die Symbole stehen bis zu drei Tastatur-Abkürzungen zur Verfügung - der in der Beschriftung unterstrichene Buchstabe, eine Funktionstaste und ggf. eine Taste auf dem Ziffernblock am rechten Rand der Tastatur. Wenn Sie (wie oben beschrieben) einen Hilfstext über eines der Symbole abrufen, zeigt HotHelp dort auch alle Tastaturkürzel des entsprechenden Symbols an.

Außerdem finden Sie im HotHelp-Projekt noch den Schlüssel 'Tastatursteuerung', unter dem sämtliche Tastatur-Bedienungsmöglichkeiten von HotHelp erläutert werden. Unter anderem erfahren Sie dort, wie der Text über die Tastatur verschoben werden kann oder wo Sie die wichtigsten Funktionen auf dem numerischen Tastenfeld wiederfinden. Sie erreichen diesen Text auch durch einen Querverweis aus demjenigen Hilfstext heraus, der bei Betätigung des Hilfe-Symbols dargestellt wird.

6.15. Möglichkeiten, HotHelp aufzurufen

Bisher wurde nur gezeigt, daß HotHelp-Fenster über die Tastenkombination <Alt>-<Help> geöffnet werden können (dies ist eine Abkürzung für 'Halten Sie die Alt-Taste gedrückt und drücken dann zusätzlich die Hilfetaste'). Wenn Sie das Fenster mehrmals schließen und über diese Kombination wieder öffnen, sehen Sie jedesmal denselben Text vor sich.

Es gibt jedoch noch drei andere Möglichkeiten, das HotHelp-Fenster zu öffnen. Dort wird dann nicht der zuletzt gezeigte Text wieder dargestellt; statt dessen bietet jede der drei Tastatur-Kombinationen die Möglichkeit, HotHelp einen bestimmten Schlüssel vorzugeben, zu dem dann Hilfe dargestellt werden soll.

Am interessantesten ist sicherlich die Kombination <Shift>-<Help>. Über sie kann Hilfe zum aktuellen Wort unter dem Cursor eines Editors abgerufen werden. Unterstützt werden die Editoren Edward von MAXON, CygnusEd von CygnusSoft und TurboText von Oxxi (voreingestellt ist der Edward, über das Programm HotHelpPref kann diese Einstellung jedoch bequem geändert werden). Dieser Tastenbefehl ist vor allem für den Programmierer extrem nützlich, da so z.B. zu jedem Begriff des Amiga-Betriebssystems (sei dies eine Funktion, eine Struktur oder eine vordefinierte Konstante) direkt und einfach Hilfe abgerufen werden kann.

Erkennt HotHelp die Tastenkombination <Ctrl>-<Help>, übernimmt das Programm das zuletzt von Ihnen eingegebene Wort und versucht, Hilfe dazu darzustellen. Dabei ist es gleichgültig, ob Sie dieses Wort in einen Editor, eine Textverarbeitung, ein Shell- oder ein HotHelp-Fenster eingegeben haben - HotHelp überwacht grundsätzlich alle Tastatureingaben und hält das jeweils zuletzt eingegebene Wort fest. Sie können dies z.B. verwenden, wenn Sie in der Shell gerade einen Befehl eingegeben haben, sich jedoch nicht mehr an die Parameter erinnern können. Da die letzte Eingabe jedoch der Name des Shell-Befehls war, können Sie über <Ctrl>-<Help> einfach einen dazu passenden Hilfstext abrufen!

Die letzte Tastenkombination <Amiga>-<Help> entnimmt den gesuchten Begriff aus dem Amiga-Zwischenspeicher (Clipboard) und öffnet ein Fenster mit einem dazu passenden Text. Ab Betriebssystemversion 2.0 können Sie diesen Hotkey z.B. verwenden, um Hilfe zu einem Wort in einem Shell-Fenster abzurufen. Markieren Sie dazu das Wort über die Maus, kopieren es über <Rechte Amiga>-<c> in den Zwischenspeicher und rufen dann über <Amiga>-<Help> Hilfe dazu ab - voila!

7. Onlineprojekte

Mit den Onlineprojekten bietet HotHelp eine faszinierende neue Möglichkeit, Texte auf einfachstem Wege in HotHelp zu integrieren. Im Gegensatz zu den normalen Projekten, deren Inhalt fest und unveränderlich ist, können Onlineprojekte jederzeit erweitert und verändert werden. Wenn Sie sich die normalen Projekte als Bücher einer Bibliothek darstellen, dann sind Onlineprojekte leere Notizbücher, in die Sie Texte eintragen, umsortieren, ändern und auch wieder löschen können. Dabei verhalten sich Onlineprojekte ganz genauso wie die normalen Projekte, so daß Sie bei der Abfrage der Texte keinerlei Unterschied feststellen werden.

7.1. Benutzung der Onlineprojekte

Um einen neuen Text zu dem aktuellen Onlineprojekt hinzuzufügen, sind nur wenige Tastendrücke erforderlich. Voraussetzung ist allerdings, daß Sie einen der drei folgenden Editoren verwenden: Edward (MAXON), CygnusEd (CygnusSoft) oder TurboText (Oxxi). Voreingestellt ist der Edward; Sie können jedoch über HotHelpPref jederzeit auch einen anderen Editor anwählen.

Benutzen Sie nun Ihren Editor, um den gewünschten Text zu schreiben. Markieren Sie den Text dann als Block und drücken anschliessend die Tastenkombination <Ctrl>-<Linke Amiga>-<s>. Dadurch wird der gerade markierte Block an HotHelp übergeben.

Sie müssen nun noch einen Schlüssel festlegen, unter dem der Text in das Onlineprojekt eingetragen werden soll, damit Sie ihn auch später wiederfinden. Dazu öffnet HotHelp ein kleines Fenster, in dem Sie sowohl den Schlüsselbegriff als auch den Namen des Onlineprojektes, in das der Schlüssel eingetragen werden soll, vorgeben können. Falls Sie mehrere Onlineprojekte verwenden, können Sie über die beiden Doppelpfeil-Symbole neben dem Projekt-Eingabefeld alle Onlineprojekte der Reihe nach durchgehen. Geben Sie einen neuen Projektnamen an, so wird das entsprechende Onlineprojekt neu angelegt.

Sind Sie mit den Einstellungen zufrieden, können Sie den neuen Schlüssel mit dem Übernehmen-Symbol in das gewählte Onlineprojekt eintragen. Wenn Sie jetzt im HotHelp-Fenster die Projektübersicht abrufen, sehen Sie dort ein weiteres Projekt mit dem Namen Ihres Onlineprojektes. Wählen Sie dieses aus, zeigt HotHelp Ihnen eine Übersicht mit allen Schlüsseln dieses Projektes an, aus der Sie dann wie üblich einen auswählen können.

Auch das Ändern eines so erstellten Textes ist ohne weiteres möglich: öffnen Sie dazu das HotHelp-Fenster und stellen Sie den gewünschten Text dar. Exportieren Sie ihn dann in Ihren Editor (eine ausführliche Anleitung zum Export von Hilfstexten finden Sie weiter oben). Sie können dazu z.B. das Clipboard- oder das Einfügen-Symbol verwenden.

Befindet der Text sich nun im Editor, können Sie ihn nach Herzenslust ändern. Anschließend können Sie ihn dann wieder (wie oben beschrieben) unter dem entsprechenden Schlüsselbegriff in das Onlineprojekt übernehmen (HotHelp ist dann jedoch mißtrauisch und fragt sicherheitshalber nach,

ob der alte Text tatsächlich durch den neuen ersetzt werden soll; beruhigen Sie es einfach, indem Sie hier mit 'Ja' antworten...).

7.2. Anwendungsgebiete für Onlineprojekte

Onlineprojekte können praktisch überall eingesetzt werden - lassen Sie Ihrer Phantasie also ruhig freien Lauf. Grundsätzlich ist ein Onlineprojekt überall da sinnvoll, wo kürzere Texte schnell festgehalten und später wieder über HotHelp abgerufen werden sollen, wobei die Texte auch möglichst leicht zu ändern sein sollen und wo auf weiter verzweigende Querverweise innerhalb der Hilfstexte verzichtet werden kann. Hier einige (mehr oder weniger) sinnvolle Vorschläge:

Programmierer können alle Funktionsprototypen und Strukturvereinbarungen in einem Onlineprojekt ablegen, so daß Sie in Zukunft auch zu jeder selbstgeschriebenen Funktion Hilfe über <Shift>-<Help> abrufen können. Ändert sich eine Funktion oder Struktur, wird der entsprechende Schlüssel einfach neu definiert. Über HotHelpPref können Sie auch die automatische Auswahl des Schlüsselbegriffs aus dem Textblock einstellen, die HotHelp ein gewisses Maß an Intelligenz verleiht und hierbei sehr hilfreich ist. Da Onlineprojekte auch keiner zahlenmäßigen Begrenzung unterliegen, können Sie sich zu jedem Programm, an dem Sie arbeiten, ein eigenes Onlineprojekt mit den Hilfstexten zu diesem Programm erstellen.

Erstellen Sie sich ein Projekt mit den Adressen aller Freunde und Bekannten, wobei Sie jeweils den Namen als Schlüsselbegriff verwenden. Änderungen (z.B. der Adresse oder der Postleitzahlen...) können Sie dann wie oben beschrieben ohne Probleme durchführen. Wenn Sie einem Bekannten einen Brief schreiben wollen, übernehmen Sie seine Adresse durch eine der zahlreichen Exportmöglichkeiten aus HotHelp in Ihre Textverarbeitung.

7.3. Onlineprojekte und andere Editoren

Auch falls Sie einen anderen als die drei genannten Editoren verwenden, brauchen Sie nicht auf die Verwendung von Onlineprojekten zu verzichten. Voraussetzung ist jedoch, daß Ihr Editor ARexx-fähig ist (s.u.). Sie müssen nur ein ARexx-Skript erstellen, das den gewünschten Textblock in das Clipboard überträgt (falls Ihr Editor das Clipboard nicht sowieso schon zum Speichern von Blöcken verwendet) und dann das Kommando HH_OnlineBlock mit entsprechenden Parametern an den Host HOTHELP_REXX sendet (weitere Informationen finden Sie im nächsten Kapitel).

8. ARexx

ARexx ist seit Version 2.0 fester Bestandteil des Amiga-Betriebssystems. Es handelt sich um eine Basic-ähnliche Sprache, deren besondere Qualität darin liegt, daß mit ihr sehr einfach andere Programme gesteuert werden können. Um mit ARexx zusammenzuarbeiten, muß ein Programm über eine sogenannte ARexx-Schnittstelle verfügen. Wie Sie sich jetzt sicher schon denken können, verfügt auch HotHelp in der neuen Version über eine solche Schnittstelle. Über diese ist es möglich, von ARexx (und damit auch von allen ARexx-fähigen Programmen aus) HotHelp-Fenster zu öffnen, zu steuern und ggf. auch wieder zu schließen.

ARexx-Interessenten finden eine komplette Übersicht über alle ARexx-Befehle von HotHelp im HotHelp-Projekt unter dem Schlüssel 'ARexx-Schnittstelle'.

Außerdem enthält HotHelp jetzt noch ein Projekt mit dem Namen 'ARexx', in dem alle Standard-Befehle und -Funktionen von ARexx erläutert werden.

9. Änderungen seit Version 2

Dieses Kapitel ist sicher für alle interessant, die HotHelp schon von der Version 2 her kennen. Es beschreibt kurz alle Änderungen, die in der neuen Version vorgenommen wurden. Eine ausführlichere Beschreibung finden Sie im vorhergehenden Kapitel oder in HotHelp selbst, wo Sie zu jedem Bedienungselement Hilfe abrufen können.

Wenn Sie Ihr erstes HotHelp-3-Fenster öffnen, wird Ihnen zuerst keinerlei Änderung auffallen. Alle Symbole befinden sich noch am selben Platz wie bisher und haben auch dieselbe Bedeutung - die einzige Ausnahme stellen die beiden Pfeilsymbole dar. Sie wurden bisher dazu verwendet, bei Eingabe eines Musterbegriffes alle passenden Schlüsselbegriffe der Reihe nach anzuzeigen. Da davon jedoch nur relativ selten Gebrauch gemacht wird, werden die Symbole in Version 3 zum Blättern durch ein Projekt verwendet - eine neue Funktion, die unter älteren Versionen noch nicht zur Verfügung stand. Die Funktionen zum Anzeigen aller zu einem Muster passenden Schlüssel wurden in das Schlüssel-Menü verschoben.

Damit sind wir auch schon beim Stichwort 'Menü' angelangt. Jedes HotHelp-Fenster verfügt jetzt über eine Menüleiste, und Sie finden hier einige der interessantesten Neuerungen von HotHelp 3: die Darstellung mehrerer Texte in einem HotHelp-Fenster (Ansichten), die leistungsstarke Suchfunktion (die auch in mehreren Texten nach einem Nicht-Schlüsselbegriff suchen kann) und die Vergabe von Lesezeichen. Außerdem können Sie von hier aus sämtliche HotHelpTools starten.

Die bisher eingebaute Hilfefunktion zu den Bedienungselementen über die rechte Maustaste war damit natürlich hinfällig. In Version 3 können Sie jedoch über das Hilfe-Symbol den Hilfsmodus aktivieren, in dem Sie durch Anklicken mit der linken Maustaste Hilfe zu den Symbolen abrufen können.

Texte können jetzt in beliebige Dateien exportiert werden, die über ein Datei-Auswahlfenster angegeben werden können. Die Beschriftung des Spezial-Symbols zeigt die aktuelle Funktion des Symbols an ('Editor' oder 'Einfügen').

Alle Symbole können jetzt (wie von Commodore empfohlen) durch Eingabe des in der Symbolbeschriftung unterstrichenen Buchstabens über die Tastatur ausgelöst werden - die bisherige, nicht standard-gemäße Kombination mit einer der Amigatasten ist nicht mehr notwendig und auch nicht mehr möglich. Bei der Auswahl eines Querverweises durch Eingabe seiner Anfangsbuchstaben muß dafür jetzt gleichzeitig eine der Alt-Tasten gedrückt werden.

Bei der Eingabe eines Musters in die beiden Eingabefelder am oberen Rand des Fensters können nun auch die aus der Shell bekannten Musterzeichen verwendet werden - vorausgesetzt, Sie arbeiten mit OS 2.0 oder höher.

Der angezeigte Hilfstext kann nun auch mit der Maus gerollt werden, indem diese bei festgehaltener linker Maustaste aus dem Textbereich herausgezogen wird.

Über die Tastenkombination <Amiga>-<Help> kann Hilfe zum aktuellen Inhalt des Zwischenspeichers (Clipboard) abgerufen werden.

HotHelp arbeitet nun auch bei <Shift>-<Help> mit dem neuen CygnusEd V3.5 zusammen - aufgrund einer inkompatiblen Änderung der ARexx-Schnittstelle dieses Editors hatte HotHelp 2 damit leider so einige Probleme...

Die Tastenkombination <Ctrl>-<Linke Amiga>-<s> übernimmt den aktuellen Block Ihres Editors und speichert ihn im aktuellen Onlineprojekt ab, wo er direkt jedem HotHelp-Fenster zur Verfügung steht. Mehr über Onlineprojekte finden Sie in Kapitel 7.

Komplett überarbeitet wurde das Programm HotHelpManager mit seiner manchmal doch etwas seltsam anmutenden Benutzerführung. Die drei Module 'Projektmanager', 'Voreinstellungen' und 'Projekte übersetzen' wurden herausgelöst und liegen nun als eigenständige Programme vor (HotHelpPro, HotHelpPref und HotHelpComp). Dazu kommen noch EasyHotHelp und der HotHelpMarker, die die Erstellung eigener Projekte extrem vereinfachen. In einer editor-ähnlichen Oberfläche können alle erforderlichen Kennungen über die Maus oder die Tastatur in die Quelltexte eingefügt werden; auch die Übersetzung, die Anzeige von Übersetzungsfehlern und der Test neuer Projekte wird mit ihm zum Kinderspiel.

Alle Programme wurden völlig neu entwickelt, wobei auch die (oftmals recht verzwickte) Benutzerführung komplett überarbeitet wurde. Vergessen Sie also am besten alles, was Sie über den alten HotHelpManager wissen und machen Sie sich mit den neuen Tools vertraut!

Falls Sie schon unter HotHelp 2 Projekte erstellt haben, hier noch einige Hinweise:

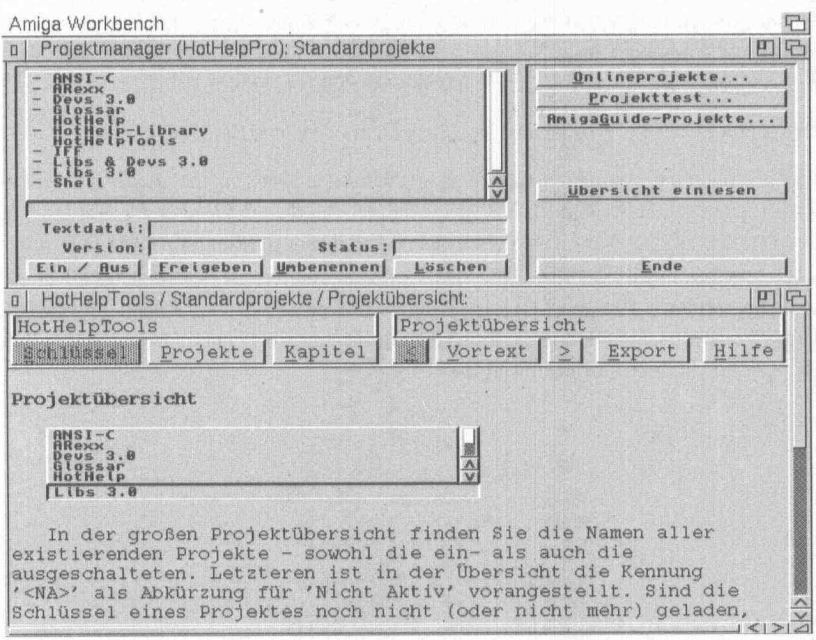
Unter HotHelp 2 konnten die einzelnen Marken bis zu 99 Zeichen lang sein; dies wurde in HotHelp 3 auf 49 Zeichen begrenzt. Außerdem (da das Suchverfahren, mit dem die Marken im Text gesucht werden, komplett überarbeitet wurde) sind überlappende Marken jetzt grundsätzlich verboten (wie z.B. 'Schlüssel' und 'SchlüsselUnsichtbar' - die erste Marke ist komplett in der zweiten enthalten). HotHelpComp überprüft alle angegebenen Marken und macht Sie ggf. vor dem Beginn der Übersetzung auf Fehler dieser Art aufmerksam. Sie müssen dann die entsprechenden Marken in den Projekt-Einstellungen und in Ihren Quelltexten anpassen.

10. Die HotHelpTools

Bisher haben Sie gesehen, wie Sie HotHelp-Fenster aufrufen und steuern können. Im folgenden wird nun auf die HotHelpTools eingegangen - dabei handelt es sich um eine Reihe von Programmen, die verschiedene Aufgaben rund um HotHelp herum übernehmen (Voreinstellungen, Wartung der Standard-, AmigaGuide- und Online-Projektdateien, Erstellung eigener Projekte).

10.1. Die Hilfsfunktion der HotHelpTools

Alle Tools verfügen über eine eingebaute, kontextsensitive Online-Hilfe - natürlich auf Basis des HotHelp-Systems. Sie gibt Ihnen Auskunft über die einzelnen Fenster der Tools, die darin auftretenden Steuerungs-Symbole und die Menüs. Sie können diese Hilfe abrufen, indem Sie den Hilfe-Menüpunkt anwählen oder (noch einfacher) die Hilfetaste drücken. Daraufhin wird ein HotHelp-Fenster geöffnet, in dem Sie einen Text über das entsprechende Tool und das gerade aktive Fenster finden. Sie können von hier aus nun über Querverweise weitere Informationen abrufen.



Noch einfacher ist es allerdings, Hilfe zu bestimmten Symbolen oder Bereichen im Fenster des Tools abzufragen. Der Mauszeiger verändert sein Aussehen, sobald Sie den Hilfsmodus aktivieren - er bekommt ein Fragezeichen angehängt. Wenn Sie mit diesem Fragezeichenzeiger auf ein Symbol im Fenster eines HotHelpTools klicken, wird in dem vorher geöffneten HotHelp-Fenster ein Hilfstext angezeigt, der die Bedeutung des angeklickten Elementes erläutert. Sie können auf diese Weise Hilfe zu allen sichtbaren Symbolen abrufen.

Um die Arbeit mit dem Tool fortzusetzen, müssen Sie zuvor den Hilfsmodus beenden. Dazu existieren zwei Möglichkeiten: Schließen Sie das HotHelp-Fenster mit den Hilfstexten oder aktivieren Sie das Fenster, von dem aus Sie den Hilfsmodus aufgerufen haben, und drücken dort nochmals die Hilfetaste. Die letztgenannte Methode hat den Vorteil, daß das HotHelp-Fenster dann geöffnet bleibt, Sie den angezeigten Hilfstext also auch weiterhin vor Augen haben.

Im den Programmen HotHelpMarker und EasyHotHelp wurde die Hilfsfunktion etwas anders gelöst, da den einzelnen Symbolen dort keine so große Bedeutung zukommt wie in den übrigen Programmen. Im Hilfsmodus wird Ihnen ein Text über das aktuelle Fenster angezeigt. Sie finden hier bereits eine Erklärung aller wichtigen Symbole vor. Auf die Möglichkeit der Auswahl eines speziellen Symbols wurde daher verzichtet.

Da in diesen beiden Programmen jedoch den Menüs (der HotHelpMarker allein bietet schon fast 100 Menüpunkte an) eine sehr wichtige Bedeutung zukommt, können Sie ab Betriebssystemversion 2.0 direkt Hilfe zu jedem Menüpunkt abrufen. Wählen Sie dazu den Menüpunkt wie üblich an. Anstatt die Maustaste dann loszulassen, drücken Sie jedoch noch zusätzlich die Hilfetaste. Das Programm öffnet dann ein HotHelp-Fenster mit einem Hilfstext über den Menüpunkt. Befindet der Mauszeiger sich über dem Menütitel oder außerhalb des Menüs, wenn die Help-Taste gedrückt wird, sehen Sie einen Hilfstext über das gesamte Menü.

Diese ausführliche Online-Hilfe ist auch der Grund dafür, daß in der folgenden Anleitung nicht auf jedes einzelne Symbol und jeden Menüpunkt eingegangen wird. Langweilige Erklärungen der Form "Um dies und das einzustellen, wählen Sie aus dem XYZ-Menü den Eintrag ABC; im sich daraufhin öffnenden Fenster klicken Sie mit der linken Maustaste auf den 0815-Schieberegler und ziehen ihn bei gedrückt gehaltener Maustaste auf die gewünschte Position. Schließen Sie das Fenster über das Verwenden-Symbol" werden Sie hier also umsonst suchen (da alleine schon das Programm HotHelpPref an die 200 Symbole beinhaltet, würde die Anleitung andernfalls bald den Umfang des neuen Postleidzahlenbuches erreichen...). Vielmehr wird im folgenden darauf eingegangen, WAS Sie mit den Programmen anfangen können - wie Sie die jeweilige Aktion dann genau durchführen, können Sie jederzeit in den Hilfstexten nachlesen!

10.2. Start der HotHelpTools

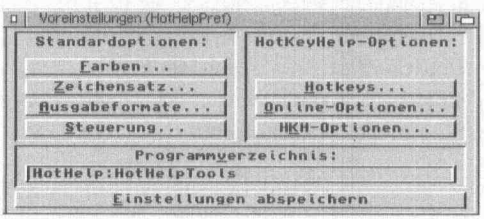
Die Programme können wie üblich von der Workbench oder der Shell aus gestartet werden. Shell-Benutzer können bei einigen Programmen außerdem beim Start zusätzliche Parameter angeben; eine Übersicht darüber erhalten Sie, wenn Sie das jeweilige Programm mit einem Fragezeichen als einzigem Parameter aufrufen. Geben Sie als Parameter 'HILFE' an, erhalten Sie einen ausführlicheren Hinweistext.

Außerdem können alle Tools aus dem HotHelpTools-Menü gestartet werden, das Sie in den Menüleisten der anderen Tools und des HotHelp-Fensters finden.

10.3. HotHelpPref - HotHelps Voreinstellungen

Wenn Sie schon mit Betriebssystem-Version 2.0 oder höher arbeiten, ist Ihnen das Prinzip der verschiedenen Voreinstellungs-Editoren sicher schon von der Workbench vertraut. Bei HotHelpPref handelt es sich ebenfalls um einen solchen Editor, mit dem Sie eine ganze Reihe (knapp 80) von verschiedenen Einstellungen des HotHelp-Systems steuern können.

Nach dem Start sehen Sie zuerst das Hauptfenster des Programms, von dem aus Sie sieben weitere Fenster öffnen können. Jedes dieser Fenster enthält Einstellungen zu einem ganz bestimmten Bereich. Die vier Symbole in der linken Spalte sind dabei für Optionen zuständig, die für alle HotHelp-Fenster gelten (also z.B. auch die, die von den Tools im Hilfsmodus geöffnet werden); die drei rechten Symbole steuern Optionen, die nur für das Programm HotKeyHelp gültig sind (dies ist das Programm, über das HotHelp-Fenster auf Tastendruck geöffnet werden können und das die Online-Projekte und die ARexx-Schnittstelle unterstützt).



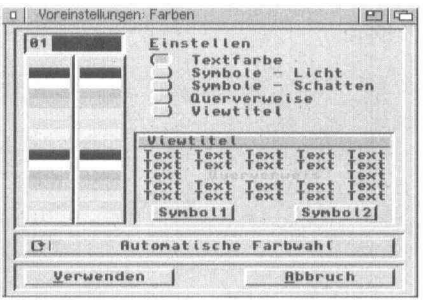
Außerdem sehen Sie hier noch ein Eingabefeld, das nur in Ausnahmefällen interessant ist (es enthält den Namen des Verzeichnisses, in dem HotHelp die HotHelpTools erwartet), sowie ein Symbol zum Abspeichern der Voreinstellungen. Nur, wenn die geänderten Einstellungen über dieses Symbol gesichert werden, werden sie von HotHelp auch verwendet.

Falls Ihr Betriebssystem noch keine verschiedenen Sprachen unterstützt, sehen Sie hier außerdem noch ein Symbol, mit dem Sie die Sprache von HotHelps Benutzerführung regeln können. Ansonsten verwendet HotHelp die eingestellte Systemsprache.

Im weiteren werden nun die Einstellmöglichkeiten, die in den verschiedenen Fenstern vorgenommen werden können, kurz erläutert. In jedem Fenster finden Sie am unteren Rand ein Verwenden- und ein Abbruch-Symbol. Nur wenn das Fenster über das Verwenden-Symbol geschlossen wird, werden die von Ihnen vorgenommenen Änderungen tatsächlich übernommen. Benutzen Sie das Abbruch-Symbol oder schließen das Fenster über das Schließsymbol, setzt HotHelpPref die Änderungen wieder zurück auf den Zustand vor dem Öffnen des jeweiligen Fensters.

10.3.1. Fenster 1: Farben

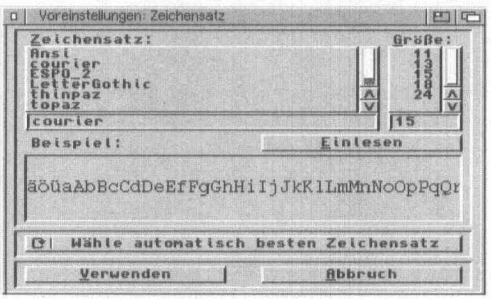
Über das Blättersymbol am unteren Fensterrand können Sie festlegen, ob HotHelp die darüber eingestellten Farben grundsätzlich verwenden soll oder ob die HotHelp-Fenster sich automatisch an die Farbzusammenstellung des verwendeten Bildschirms anpassen sollen. Normalerweise ist die zweite Methode vorzuziehen.



Falls Sie sich für die Vorgabe von Farben entscheiden, können Sie im darüberliegenden Bereich Farben für den Text, die helle und die dunkle Kante der Symbole, die Querverweise und den Titel der aktiven Ansicht angeben.

10.3.2. Fenster 2: Zeichensatz

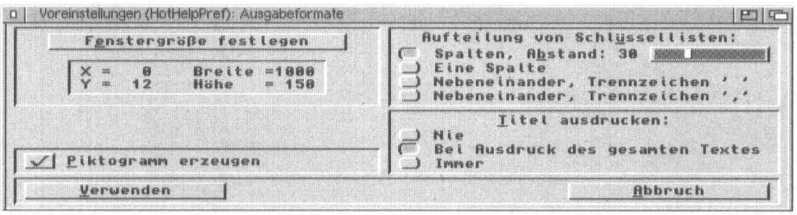
Ähnlich wie im Farbenfenster können Sie auch hier über ein Blättersymbol einen festen Zeichensatz vorgeben oder HotHelp anweisen, den Zeichensatz automatisch an die aktuelle Bildschirmauflösung anzupassen. Daneben können Sie auch noch festlegen, daß beim Öffnen immer der Zeichensatz des zuletzt aktiven Fensters übernommen werden soll. Wenn Sie sich bei der Installation für eine Version mit beschleunigter Textausgabe entschieden haben, steht Ihnen noch eine weitere entsprechende Option zur Verfügung.



Im Bereich über dem Blättersymbol können Sie einen Zeichensatz auswählen. Je nach Einstellung wird dieser grundsätzlich verwendet oder nur in die automatische Entscheidung mit einbezogen. Außerdem sehen Sie immer eine Schriftprobe des aktuellen Zeichensatzes.

10.3.3. Fenster 3: Ausgabeformate

In diesem Fenster können Sie alle Optionen editieren, die sich irgendwie mit der Ausgabe von Hilfstexten befassen.



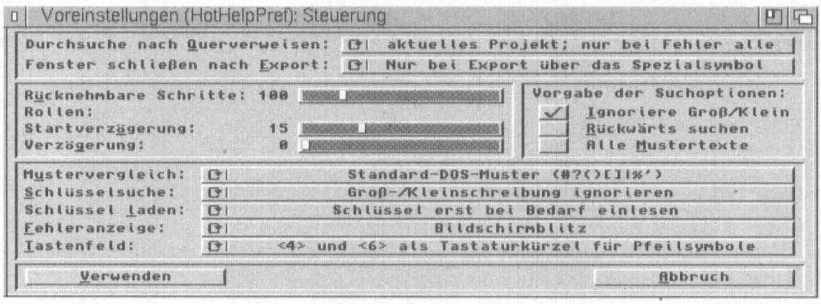
Dazu zählt als erstes die Größe und Position des Fensters. Diese Werte werden immer verwendet, wenn Sie nach einem Neustart des Rechners das erste HotHelp-Fenster öffnen.

Als zweites können Sie Einfluß auf die Aufteilung von Schlüssellisten nehmen. Es handelt sich dabei um Übersichten von Schlüsseln, die HotHelp z.B. bei Eingabe eines Musters erzeugt. Auch in normalen Hilfstexten können solche Listen auftreten. Diese Einstellung ist letzten Endes ein reine Geschmacksfrage.

Die darunter liegenden Optionen befassen sich mit dem Export von Hilfstexten. Sie können dort angeben, ob beim Export in eine Datei ein Workbench-Piktogramm erzeugt werden soll (reine Shell-Benutzer werden diese Option sicherlich gerne ausschalten) und unter welchen Bedingungen dem Ausdruck eines Hilfstextes sein Titel vorangestellt werden soll.

10.3.4. Fenster 4: Steuerung

In diesem Fenster mit seinem allgemeinen Namen wurden letztlich alle Optionen untergebracht, die sich keinem anderen Fenster mehr zuordnen ließen.



Im obersten Feld können Sie den Bereich festlegen, in dem die Suche nach Querverweisen durchgeführt werden soll - entweder nur im aktuellen Projekt, nur zuerst im aktuellen Projekt oder grundsätzlich in allen Projekten.

Darunter wird bestimmt, in welchen Fällen das HotHelp-Fenster nach dem Export geschlossen werden soll.

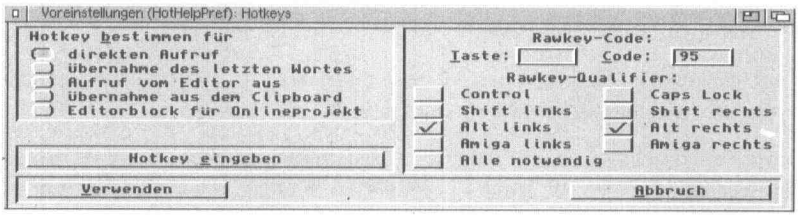
Über die drei Schieberegler können Sie die Anzahl der über das Vortext-Symbol rücknehmbaren Schritte und die Verzögerung beim Rollen über die Maus oder die Rollsymbole im Fensterrahmen einstellen.

Im daneben liegenden Feld können Sie die Suchoptionen festlegen, die beim ersten Öffnen eines HotHelp-Fensters eingestellt sein sollen.

Über die fünf Blättersymbole am unteren Fensterrand können Sie festlegen, ob Sie in den Eingabefeldern die normalen Shell-Muster (erst verfügbar ab Betriebssystemversion 2.0) oder die einfachen HotHelp-Muster verwenden wollen, ob die Groß-/Kleinschreibung eines Begriffs bei der Schlüsselsuche beachtet werden soll, ob die Schlüsselübersichten erst bei Bedarf nachgeladen und bei Speichermangel wieder freigegeben werden dürfen, ob kleine Bedienungsfehler mit einem Bildschirmblitz, einem Piepton, beidem oder nichts von beidem quittiert werden sollen und wie die Tasten 4 und 6 auf dem numerischen Tastenblock verwendet werden sollen.

10.3.5. Fenster 5: Hotkeys

In diesem Fenster können Sie die fünf Tastenkombinationen editieren, die als Hotkeys abgefragt werden sollen. Dies ist sehr sinnvoll, wenn sich eine der HotHelp-Tastenkombinationen mit der eines anderen Programms überschneidet. Sie können hier die entsprechende Kombination ändern.

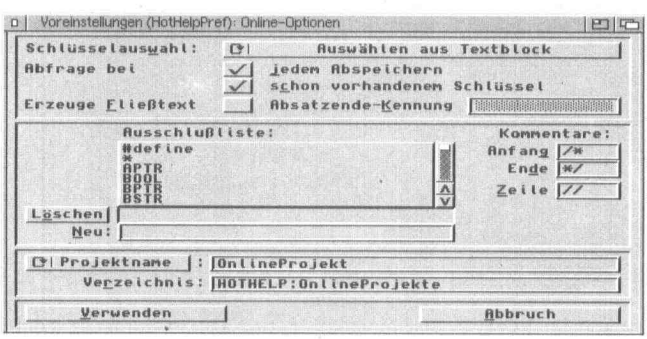


Über die fünf Druckknopfsymbole wählen Sie zuerst aus, welchen der Hotkeys Sie editieren wollen. Im großen Feld rechts können Sie dann den Rawkey-Code und die Qualifier-Kombination einstellen, die für den Hotkey gelten soll. Einfacher ist es allerdings, über das Symbol 'Hotkey eingeben' ein weiteres Fenster zu öffnen, in dem Sie dann die gewünschte Tastenkombination direkt eingeben können, die als neuer Hotkey verwendet werden soll.

10.3.6. Fenster 6: Online-Optionen

In diesem Fenster sind alle Optionen zusammengefaßt, die sich auf Onlineprojekte beziehen.

Im oberen Feld können Sie zuerst die Art und Weise festlegen, auf die der Schlüssel bestimmt werden kann, unter dem ein neuer Text eines Onlineprojektes abgelegt werden soll. Sie können sich entscheiden zwischen Eingabe per Hand, Auswahl aus dem selektierten Block oder automatischer Auswahl.



Bei der letzten Option wird die Ausschlußliste verwendet, die Sie darunter editieren können. HotHelp untersucht dann den Textblock Wort für Wort und benutzt das erste Wort als Schlüssel, das nicht in der Ausschlußliste auftritt. Diese Option ist sehr sinnvoll, wenn Sie z.B. Funktionsprototypen in Online-Projekte aufnehmen (voreingestellt sind die Worte, die bei C-Programmen über-

sprungen werden sollen - Sie können die Liste jedoch jederzeit an Ihre Programmiersprache anpassen).

Weiterhin können Sie festlegen, ob eine Sicherheitsabfrage bei jedem Anlegen eines neuen Textes bzw. vor dem Überschreiben eines schon vorhandenen Textes durchgeführt werden soll. Außerdem kann für den Fall, daß die Onlineprojekte als Fließtext übersetzt werden sollen, eine Marke zur Kennzeichnung von Absatzenden definiert werden.

Neben der Ausschlußliste können Sie auch das Format von Kommentaren angeben. Worte in Kommentaren werden bei der automatischen Schlüsselauswahl nicht berücksichtigt.

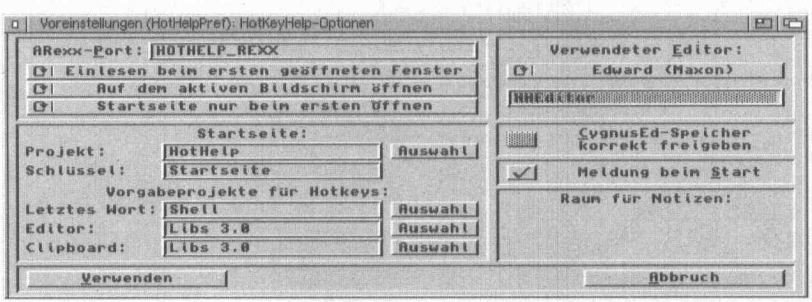
Darunter können Sie schließlich noch den Namen des Onlineprojektes eingeben, in das neue Texte vorgabemäßig eingetragen werden. Ab Betriebssystemversion 2.0 ist auch die Angabe des Namens einer Umgebungsvariablen möglich; HotHelp entnimmt dann beim Einfügen eines neuen Textes den Namen des gewünschten Onlineprojektes aus dieser Variablen.

Das Verzeichnis, in dem Onlineprojekte abgelegt werden, kann ebenfalls hier verändert werden, auch wenn dies im allgemeinen weder notwendig noch ratsam ist.

10.3.7. Fenster 7: HKH-Optionen

Ähnlich wie im Steuerungsfenster sind hier alle übrigen HotKeyHelp-Optionen (HKH-Optionen) zusammengefaßt, die sich jeglicher Zuordnung zu anderen Fenstern entzogen...

Im ersten Feld können Sie den ARexx-Namen festlegen, unter dem HotHelp angesprochen werden kann. Darunter sehen Sie drei Blättersymbole, mit denen Sie bestimmen können, zu welchem Zeitpunkt HotHelp seine Daten einliest, ob das Fenster sich nur auf Public-Bildschirmen öffnen darf und wann die eingestellte Startseite angezeigt werden soll.



Im darunterliegenden Bereich können Sie die Startseite vorgeben, die beim ersten (oder jedem) Öffnen eines Fenster über die Tastenkombination <Alt>-<Help> angezeigt werden soll. Darunter können Sie Vorgabeprojekte für die drei übrigen Hotkeys angeben. Wird der entsprechende Hotkey erkannt, sucht HotHelp den gesuchten Begriff zuerst in dem hier eingestellten Vorgabeprojekt. Über das Auswahl-Symbol können Sie ein Fenster mit einer Projektübersicht öffnen, aus der Sie das gewünschte Projekt auswählen können.

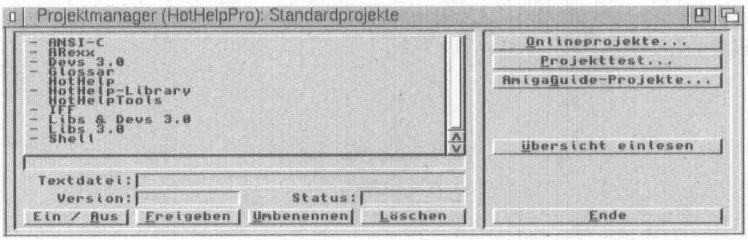
Wenn Sie über <Shift>-<Help> Hilfe zum aktuellen Wort Ihres Editors abrufen oder die Online-Projekte verwenden wollen, ist es notwendig, im Eingabefeld oben rechts den von Ihnen verwendeten Editor einzustellen. Ab Betriebssystemversion 2.0 können Sie auch den Namen einer Umgebungsvariablen angeben, aus der HotHelp bei jeder Verwendung den Namen des aktuellen Editors entnehmen soll.

Das Feld darunter ist nur für Benutzer des CygnusEd vor Version 3.5 interessant, da bei diesen Versionen Speicherverluste bei der Kommunikation mit HotHelp auftreten. Ist das Auswahlfeld aktiviert, korrigiert HotHelp diese Fehler automatisch.

Das letzte Feld schließlich bestimmt, ob HotKeyHelp beim Start kurz ein Fenster mit einer Nachricht öffnen soll oder nicht.

10.4. HotHelpPro - Der Projektmanager

HotHelps Projektmanager enthält vier Module, die alle für die Arbeit mit den verschiedenen Arten von Projektdateien zuständig sind: die Pflege der Standardprojekte (Ein-/Ausschalten, Freigeben von Speicher, Verschieben, Löschen), die Verwaltung der Onlineprojekte, An- und Abmelden von AmigaGuide-Dateien als HotHelp-Projekte sowie die Überprüfung von selbsterstellten Projekten auf Vollständigkeit und logische Fehler. Jedes dieser Module verfügt über ein eigenes Fenster, das aus jedem der anderen Module heraus angesprochen werden kann. Beim Start wird üblicherweise das Standardprojekte-Fenster geöffnet.



10.4.1. Standardprojekte

Den größten Teil dieses Fensters nimmt eine Liste ein, in der Sie die Namen aller vorhandenen Standardprojekte sehen können. Wählen Sie eines der Projekte durch Anklicken aus, zeigt das Programm Ihnen in den Feldern unter der Liste einige Informationen über das Projekt an. Sie sehen dort den Namen der Textdatei des Projektes, seine Versionsnummer und eine Angabe, ob das Projekt ein- oder ausgeschaltet ist.

Über die vier Symbole unterhalb der Liste können Sie nun das ausgewählte Projekt beeinflussen. Das erste Symbol schaltet das Projekt ein bzw. aus. Dies ist sehr nützlich, wenn Sie ein Projekt längere Zeit nicht benötigen. Sie können es dann hier ausschalten, so daß es keinen Speicherplatz verbraucht und auch in der Projektübersicht nicht mehr aufgeführt wird. Wenn Sie dann wieder auf das Projekt zugreifen wollen, können Sie es hier wieder einschalten. Ausgeschalteten Projekten wird in der Liste das Zeichen '0' vorangestellt.

Das 'Freigeben'-Symbol gibt den Speicher frei, den das entsprechende Projekt für die Übersicht über seine Schlüssel angelegt hat, ohne jedoch das gesamte Projekt aus dem Speicher zu entfernen. Sie erkennen dies daran, daß dem Projekt in der Übersicht ein Minuszeichen vorangestellt wird.

Sind Sie sich sicher, daß Sie ein Projekt nie wieder benötigen, können Sie es über das entsprechende Symbol auch komplett löschen. Aber Vorsicht: Einmal gelöschte Dateien sind für immer verloren!

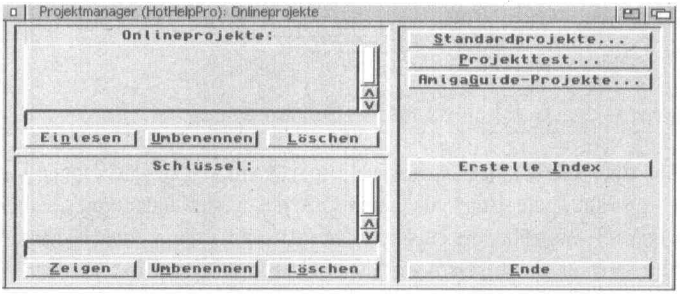
Mit dem Symbol 'Umbenennen' können Sie die Textdatei des Projektes umbenennen oder in ein anderes Verzeichnis verschieben. Dazu eine kleine Erklärung: HotHelps Projekte bestehen aus zwei Dateien; die Kopfdatei mit der Endung ".hdr" enthält Angaben über die Schlüssel des Projektes und

muß sich immer im Verzeichnis "HOTHELP:Projekte" befinden. Die Textdatei mit der Endung ".txt" enthält alle Hilfstexte des Projektes und kann in einem beliebigen Verzeichnis liegen. Sie dürfen die Textdatei allerdings keinesfalls einfach über die Workbench oder die Shell verschieben oder umbenennen, sondern Sie müssen dazu unbedingt dieses Symbol des Projektmanagers benutzen, damit HotHelp nachvollziehen kann, wo sich die Datei befindet.

Über die vier Symbole am rechten Fensterrand können Sie die anderen drei Module abrufen oder das Fenster schließen.

10.4.2. Onlineprojekte

In diesem Fenster können Sie die Texte aller definierten Onlineprojekte bearbeiten. Sie haben die Möglichkeit, ganze Onlineprojekte oder auch einzelne Schlüssel daraus umzubenennen oder ganz zu löschen.

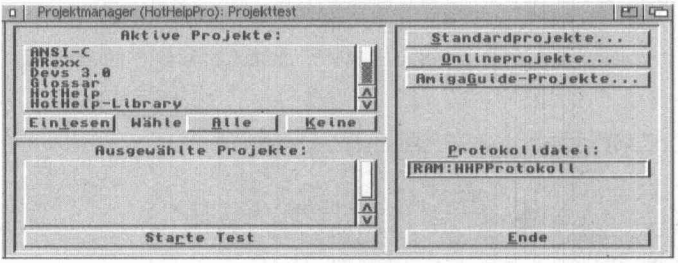


Oben links sehen Sie dazu eine Liste aller Onlineprojekte, aus der Sie das gewünschte Projekt auswählen können. Daraufhin wird in der unteren Liste eine Übersicht über alle Schlüssel dieses Projektes angezeigt. Über das obere Umbenennen- und Löschen-Symbol können Sie nun das ausgewählte Onlineprojekt bearbeiten; wollen Sie nur einen Schlüssel des Onlineprojektes manipulieren, können Sie diesen aus der unteren Liste auswählen und dann umbenennen oder löschen. Um sicherzugehen, können Sie sich den zum angewählten Schlüssel gehörenden Text auch zuvor in einem HotHelp-Fenster anzeigen lassen.

Das Symbol 'Erstelle Index' ist nur für Notfälle vorgesehen, wenn durch einen Diskettenfehler die Indexdatei der Onlineprojekte zerstört wurde (HotHelp hält in dieser Datei Verwaltungsinformationen über alle Onlineprojekte fest). In einem solchen Fall können Sie über dieses Symbol die Indexdatei neu erstellen lassen.

10.4.3. Projekttest

Dieses Modul ist dann für Sie interessant, wenn Sie selbst HotHelp-Projekte erstellen möchten. Es prüft ein oder mehrere Projekte auf logische Fehler. Dazu untersucht das Programm alle definierten Schlüssel und Querverweise des Projektes. Falls zu einem Querverweis ein passender Schlüssel nur in einem anderen Projekt auftaucht oder wenn gar kein passender Schlüssel existiert, zeigt das Programm eine Warnung an. Weiterhin wird noch eine Meldung erzeugt, wenn in einem Projekt ein Schlüssel mehrfach definiert wird - auch dies kann auf einen Fehler hindeuten. Auf diese Weise lassen sich recht einfach schwerwiegende Fehler bei der Erstellung eines Projektes abfangen (z.B. durch falsch geschriebene Querverweise oder ganz einfach vergessene Schlüssel).

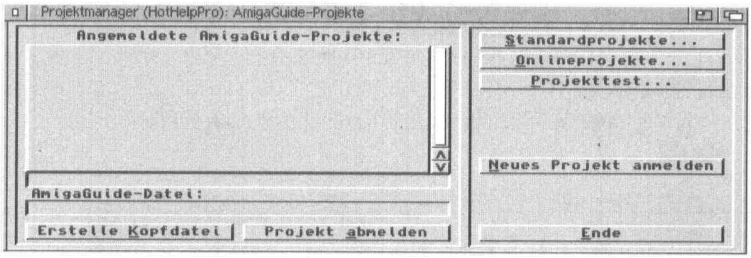


Die obere Liste enthält eine Übersicht über alle eingeschalteten Standardprojekte. Sie können hier ein oder mehrere Projekte auswählen, die dann in die untere Liste der ausgewählten Projekte übertragen werden. Wenn Sie den Test dann über das entsprechende Symbol starten, werden alle ausgewählten Projekte der Reihe nach untersucht. Dazu wird ein eigenes Fenster geöffnet, in dem Sie den Fortschritt des Testvorgangs verfolgen können. Alle gefundenen Fehler werden in einer Liste angezeigt. Außerdem werden alle Fehlermeldungen zusätzlich in eine Protokolldatei eingetragen, die Sie nach Abschluß des Tests durchlesen können.

10.4.4. AmigaGuide-Projekte

AmigaGuide ist das von Commodore favorisierte Hilfssystem, das seit OS 3.0 auch mit dem Betriebssystem zusammen ausgeliefert wird. AmigaGuide hat jedoch zwei große Probleme: zum einen ist er noch lange nicht für jeden Anwender verfügbar, da nicht jeder über OS 3.0 oder höher verfügt; zum anderen bietet er keinen besonderen Komfort - er ist nicht viel mehr als ein verbessertes Textanzeigeprogramm und kann in der aktuellen Version noch nicht einmal vernünftig über die Tastatur gesteuert werden.

Das AmigaGuide-Format ist aber dennoch interessant, da eine wachsende Anzahl von Programmen (vor allem in der Public Domain, aber auch einige kommerzielle Anwendungen) über Dokumentationen im AmigaGuide-Format verfügt. Aus diesem Grund können nun auch AmigaGuide-Dateien als normale Projektdateien in das HotHelp-System eingebunden werden. Sie können dann beim Ansehen dieser Texte den vollen Komfort ausnutzen, den HotHelp Ihnen bietet.

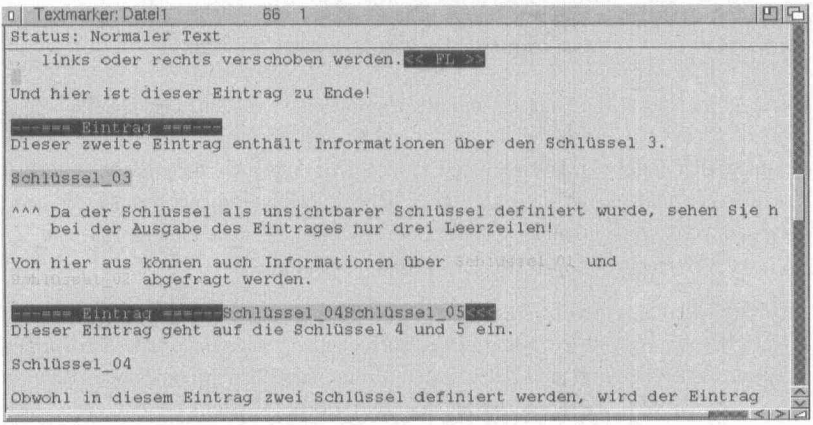


Dieses Modul dient nun dazu, eine AmigaGuide-Datei als HotHelp-Projekt anzumelden. Sie können über das Symbol 'Neues Projekt anmelden' ein Datei-Auswahlfenster öffnen, in dem Sie die gewünschte AmigaGuide-Datei auswählen können. HotHelpPro untersucht dann die Datei, erstellt eine Schlüsselübersicht und meldet die Datei unter dem in ihr angegebenen Namen als neues HotHelp-Projekt an. Von jetzt an können Sie in jedem HotHelp-Fenster auf diesen AmigaGuide-Text zugreifen, als ob es sich um ein normales HotHelp-Projekt handeln würde.

In der großen Liste sehen Sie eine Übersicht über alle bereits angemeldeten AmigaGuide-Projekte. Wählen Sie eines davon aus, wird der Name der entsprechenden AmigaGuide-Datei in dem Feld darunter angezeigt. Sie können nun mittels der beiden darunterliegenden Symbole das Projekt wieder abmelden (dadurch wird die Datei wieder aus dem HotHelp-System entfernt, aber nicht gelöscht) oder die Kopfdatei neu erstellen. Diese Aktion ist nur notwendig, wenn die AmigaGuide-Datei seit der Anmeldung noch geändert wurde, was normalerweise nicht der Fall sein sollte. HotHelp macht Sie jedoch automatisch darauf aufmerksam, wenn die Kopfdatei neu erstellt werden muß.

10.5. HotHelpMarker - der Projekte-Editor

Dies ist das umfangreichste der HotHelpTools, mit dessen Hilfe Sie ohne weiteres eigene Projekte mit Hilfstexten zu beliebigen Themen anfertigen können. Sie finden dazu im folgenden zuerst einen Abschnitt über die Grundlagen der Entwicklung von Hilfstexten, bevor dann auf die Bedienung des Markers eingegangen wird.



10.5.1. Grundlagen der Projekterstellung

Beim Aufbau von Hilfstexten gibt es zwei grundsätzlich verschiedene Möglichkeiten: Interpretieren und Kompilieren. Im ersten Fall handelt es sich bei den Hilfstexten um ganz normale ASCII-Texte, in dem bestimme Bereiche besonders markiert werden (z.B. als Schlüssel oder Querverweise). Wird ein solcher Hilfstext dargestellt, muß er nach dem Einlesen zuerst auf diese Markierungen hin untersucht werden, bevor der Text angezeigt werden kann. Diese Methode hat den Vorteil, daß die Texte schnell änderbar sind und auch ohne das spezielle Hilfsprogramm angezeigt werden können. Nachteilhaft sind jedoch die längere Wartezeit beim Einlesen und die Tatsache, daß Texte nicht komprimiert werden können, da sie ja im ASCII-Format vorliegen müssen.

Beim Kompilieren werden die Hilfstexte ebenfalls als ASCII-Texte mit besonders markierten Bereichen erstellt. Diese Texte (die als Quelldateien bezeichnet werden, da sie den Ausgangspunkt der Hilfsdateien darstellen) werden dann jedoch von einem Übersetzer in ein Binärformat umgesetzt. Die Texte können dann zwar nur noch mit dem jeweiligen Hilfsprogramm dargestellt werden; der Vorteil dieser Methode liegt jedoch in den kürzeren Ladezeiten der einzelnen Texte (die Suche nach den Markierungen wurde ja schon bei der Übersetzung durchgeführt) und der Möglichkeit, die Texte zu komprimieren, wodurch viel Platz eingespart werden kann.

Da diese Vorteile (kürzere Ladezeit und um durchschnittlich 50 % reduzierter Platzbedarf) die Nachteile weit übertreffen, wurde für HotHelp die Methode der kompilierten Hilfstexte gewählt.

Die grundlegenden Schritte bei der Erstellung eines Projektes sehen also so aus:

- ☞ Schreiben der Hilfstexte (Quelltexte)
- ☞ Markieren bestimmter Textbereiche als Schlüssel, Querverweise, ...
- ☞ Übersetzen der mit Markierungen versehenen Texte

Sie finden dazu im folgenden eine Einführung in die Erstellung von Projekten, die Ihnen Schritt für Schritt erklärt, wie Sie Hilfsdateien mit eigenen Texten erstellen können. Dabei werden nur die wichtigsten Menüs und Bedienungselemente erläutert, die für die jeweiligen Arbeitsschritte benötigt werden. Informationen über die restlichen Bedienungselemente können Sie jedoch jederzeit (wie weiter oben beschrieben) über die Help-Taste abrufen.

Während der Installation haben Sie die Möglichkeit, die Quelltexte von zwei Beispielprojekten zu installieren. Das erste Beispielprojekt beschreibt mit ausführlichen Kommentaren den Aufbau von Quelltexten und die Verwendung der verschiedenen Marken, während das zweite die sinnvolle Benutzung von Schlüsselgruppen demonstriert. Vorgabemäßig werden die Beispieldateien im Verzeichnis 'Work:HotHelpDemoProjekte' in zwei Unterverzeichnissen abgelegt. Wenn Sie die folgenden Kapitel durcharbeiten, können Sie jederzeit über den HotHelpMarker die Quelltexte der beiden Beispielprojekte darstellen, um einige praktische Beispiele für die Ausführungen dieser Anleitung zu erhalten.

10.5.2. Hintergrund

Fangen Sie am besten damit an, sich Gedanken über das Thema Ihres Projektes zu machen. Überlegen Sie sich, ob (und wenn ja, wie) Sie die Informationen in Teilbereiche aufteilen können. Falls eine solche Aufteilung sinnvoll ist, sollten Sie Ihr Projekt in Kapitel (und ggf. Unterkapitel) aufteilen. HotHelp unterstützt den Aufbau von Kapitelebenen mit beliebiger Tiefe. Machen Sie sich ggf. eine Skizze, aus der der spätere Aufbau des Projektes hervorgeht. Dies ist um so wichtiger, je komplexer der Aufbau des Projektes sein soll - wenn Sie z.B. eine Aufteilung in Kapitel, Unterkapitel etc. vornehmen wollen, sollten Sie sich schon von Anfang an darüber klar sein, wie diese Struktur aufgebaut sein soll. Spätere Änderungen der Struktur sind meist sehr aufwendig und bergen das Risiko von logischen Fehlern im Kapitelaufbau.

Verfeinern Sie nun die vorgenommene Aufteilung, bis sich einzelne, nicht weiter teilbare Gebiete herauskristallisieren. Diese Bereiche können Sie sich als die späteren Einträge des Projektes vorstellen - Texte, denen sich ein oder mehrere eindeutige Schlüsselbegriffe zuordnen lassen und die alle Informationen über den entsprechenden Begriff beinhalten.

10.5.3. Eingabe der Texte

Nachdem Sie sich über den Aufbau Ihres Projektes klar geworden sind, sollten Sie die entsprechenden Texte (die sog. Quelltexte) erstellen. Sie können dazu einen beliebigen Editor oder eine Textverarbeitung verwenden (letztere muß Texte im üblichen ASCII-Format abspeichern können).

Die Quelldateien

Ein Projekt kann sich aus beliebig vielen Dateien mit Quelltexten zusammensetzen. Sie können also alle Texte eines Projektes in einer Datei zusammenfassen oder z.B. Dateien anhand der weiter oben beschriebenen logischen Gliederung bilden, wobei sich in jeder Datei alle Texte zu dem entsprechenden Teilgebiet befinden. Für kleine Projekte ist es sicher am einfachsten, alle Texte in einer Datei zusammenzufassen, während bei großen Projekten bei dieser Methode wahrscheinlich schnell die Übersichtlichkeit verloren geht. Entscheiden Sie sich selbst für den für Ihre Zwecke besten Weg!

Die Dateinamen sind ebenfalls beliebig; allerdings müssen Sie zwei Besonderheiten beachten: Die Namen dürfen nicht länger als 26 Zeichen sein und keine Datei darf unter dem Namen 'HotHelp-Compiler.Pref' oder 'HotHelpCompiler.Titles' abgelegt werden - diese beiden Dateien werden dazu verwendet, HotHelp-interne Daten zu den Quelldateien jedes Projektes festzuhalten.

Alle Quelldateien des Projektes müssen unbedingt in einem einzigen Verzeichnis zusammengefaßt werden. In diesem Verzeichnis sollten sich auch keine Quelltexte anderer Projekte befinden; das Verzeichnis muß ausschließlich für die Quelldateien dieses einen Projektes reserviert sein! Am einfachsten ist es, wenn Sie dem Verzeichnis den Namen geben, den auch Ihr Projekt später tragen soll.

Wenn Sie eine Festplatte besitzen, sollten Sie sich ein eigenes Verzeichnis auf der Platte anlegen (z.B. 'WORK:HotHelpSource'), in dem Sie dann alle Unterverzeichnisse mit den Quelldateien Ihrer verschiedenen Projekte ablegen können. Wenn Sie dann noch die Zeile 'Assign HHC: WORK:HotHelpSource' in Ihre Startup-Sequence einsetzen, können Sie später vom HotHelp-Textmarker aus über das HHC:-Aktionssymbol des Dateiauswahl-Dialogs schnell Ihre Quelldateien erreichen.

Diskettenbenutzer sollten eine eigene Diskette für die Quelldateien der Projekte verwenden, auf der dann die einzelnen Verzeichnisse für die verschiedenen Projekte angelegt werden können. Geben Sie der Diskette am besten den Namen 'HHC', dann können Sie (wie oben beschrieben) schnell auf die Verzeichnisse zugreifen.

(Wie Sie Verzeichnisse anlegen, neue Disketten formatieren und Disketten umbenennen können, entnehmen Sie bitte Ihrem Amiga-Benutzerhandbuch).

Die Texte

Geben Sie nun die Texte ein. Sie können sich dabei ganz auf die Inhalte konzentrieren - das Markieren von Bereichen muß nun nicht mehr (wie in der vorherigen HotHelp-Version) von Hand durch Einfügen von mehr oder weniger einprägsamen Kennungen in den Text durchgeführt werden - dafür wird im nächsten Schritt der HotHelp-Textmarker herangezogen!

10.5.4. Markieren der Texte

Der HotHelp-Textmarker

Starten Sie nun bitte den HotHelp-Textmarker. Rufen Sie als erstes aus dem Projekt-Menü den Punkt 'Projektverzeichnis' auf, um das Verzeichnis festzulegen, mit dem Sie arbeiten wollen. Wählen Sie dazu im Datei-Auswahlfenster das Verzeichnis aus, in dem sich Ihre eben erstellten Quelldateien

befinden und bestätigen Sie die Auswahl über 'OK'. Da das neue Verzeichnis noch keine Voreinstellungs-Datei beinhaltet, werden Sie gefragt, ob eine solche Datei mit Vorgabeeinstellungen angelegt werden soll. Beantworten Sie die Frage mit 'Ja'. Nun können Sie über den Menüpunkt 'Öffnen' aus dem Projekt-Menü eine Ihrer Quelldateien öffnen. Sie wird im Editorfenster dargestellt.

Während der Bearbeitung Ihrer Texte öffnen sich ggf. eine Reihe von Dialogfenstern, in denen Sie bestimmte Eingaben vornehmen müssen. Drücken Sie in einem dieser Fenster die Hilfetaste, so öffnet das Programm ein HotHelp-Fenster mit einem Hilfstext über das jeweilige Fenster, in dem alle Bedienungselemente erklärt werden.

Aufteilung des Textes

HotHelp bietet die Möglichkeit, die Hilfstexte als Fließtext zu verwalten. Dabei müssen bei der Erstellung der Texte jedoch einige Regeln beachtet werden. Ist Ihnen das zu umständlich, können Sie die Fließtext-Übersetzung auch ausschalten. Wählen Sie dazu im Voreinstellungen-Untermenü des Projekt-Menüs den Punkt 'Editieren' an und schalten Sie im daraufhin erscheinenden Optionen-Fenster das Auswahlfeld 'Erzeuge Fließtext' aus. Wenn Sie diese Einstellung verwenden, behält HotHelp bei der Übersetzung die zeilenweise Aufteilung Ihrer Quelltexte bei. Die Texte werden dann genau so ausgegeben, wie Sie sie im Quelltext angelegt haben. Ist eine Zeile dann allerdings zu lang, muß der Leser den sichtbaren Textausschnitt über die Rollsymbole von Hand nach rechts verschieben, wenn er das Zeilenende lesen möchte.

Wenn Sie Ihre Hilfstexte als Fließtext übersetzen lassen, faßt HotHelp alle direkt aufeinander folgenden Zeilen zu Absätzen zusammen. Absätze werden beendet durch Leerzeilen oder durch die Absatzende-Kennung. Sie müssen nun also entweder zwischen allen Absätzen Leerzeilen einfügen oder (wenn zwei Absätze ohne Leerzeile aufeinander folgen sollen) die Absatzende-Kennung hinter dem ersten der beiden Absätze einfügen. Bewegen Sie dazu den blinkenden Cursor hinter das Ende des ersten der beiden Absätze, die nicht durch eine Leerzeile getrennt werden sollen. Wählen Sie dann aus dem Bearbeiten-Menü im Untermenü 'Kennung einfügen' den Punkt 'Absatzende' (alternativ können Sie auch die Tastenkombination <Alt>-<F4> benutzen). Vor dem Cursor erscheint der Text '<<<', der farblich hervorgehoben ist. Sie haben damit hinter dem ersten Absatz eine Absatzende-Kennung eingefügt.

In den meisten Fällen kann allerdings auf diese Kennung verzichtet werden, wenn Leerzeilen zwischen den Absätzen eingefügt werden.

Einteilung in Einträge

Als nächstes sollten Sie nun die einzelnen Einträge (das sind die einzeln darstellbaren Hilfstexte) innerhalb der Datei voneinander abgrenzen. Hierzu gehen Sie genauso vor wie beim Einfügen der Absatzende-Kennung: plazieren Sie den Cursor ans Ende des ersten Eintrags und wählen dann im Untermenü 'Kennung einfügen' den Punkt 'Eintragsende' an. Es erscheint der farblich hervorgehobene Text '—=== Eintrag ===—', der ein Eintragsende symbolisiert. Fügen Sie nun auch hinter allen weiteren Einträgen die Kennungen ein (auch hinter dem letzten Eintrag, also am Ende der Datei!). Damit haben Sie den ersten wichtigen Schritt vollzogen: der Text der Datei wurde in eine Reihe unabhängiger Einträge aufgeteilt.

Die Schlüssel

Als nächstes muß jedem Eintrag mindestens ein eindeutiger Schlüsselbegriff zugewiesen werden, unter dem der Eintrag später angesprochen werden kann. Bewegen Sie dazu den Cursor an den Anfang der Datei. Wählen Sie nun aus dem Bearbeiten-Menü im Untermenü 'Bereichsmarken' den Punkt 'Schlüssel' an. Der HotHelp-Textmarker zeigt jetzt in der Titelzeile des Fensters an, daß die Schlüssel-Marke gesetzt ist. Suchen Sie nun im ersten Eintrag nach dem Begriff, der als Schlüssel für den Eintrag verwendet werden soll. Meist taucht dieser Begriff schon in der Überschrift des Eintrags auf (falls er eine solche hat; andernfalls können Sie jetzt noch eine eingeben). Positionieren Sie den Cursor dann auf das erste Zeichen des gewünschten Schlüsselbegriffs und wählen aus dem Bearbeiten-Menü den Menüpunkt 'Bereich markieren' und daraus den Untermenüpunkt 'Blockanfang'. Bewegen Sie den Cursor dann auf das letzte Zeichen des Schlüsselbegriffs und wählen aus demselben Untermenü 'Blockende'. Der so markierte Begriff wird farblich hervorgehoben.

Gehen Sie jetzt alle Einträge durch und weisen Sie jedem Eintrag mindestens einen Schlüssel zu. Enthält ein Eintrag keinen Begriff, der als Schlüssel passend wäre, können Sie natürlich nachträglich noch einen eingeben. Ebenso steht es Ihnen natürlich auch frei, noch weitere Änderungen am Text vorzunehmen. Bei größeren Änderungen sollten Sie allerdings die Datei abspeichern und in Ihren Editor laden, da der HotHelp-Textmarker nicht als Editor konzipiert wurde und daher auch keine umfangreichen Editiermöglichkeiten bietet. Zur Korrektur von Tippfehlern oder Eingabe kleinerer Änderungen eignet er sich allerdings so gut wie ein richtiger Editor.

Eine besondere Rolle nimmt noch der Schlüssel 'Startseite' ein. Wird ein Projekt aus der Projektübersicht ausgewählt, sucht HotHelp als erstes einen Schlüssel mit dem Namen 'Startseite' in dem gewählten Projekt. Wird er gefunden, stellt das Programm den zugehörigen Eintrag dar. Existiert kein solcher Schlüssel, wird statt dessen eine Übersicht über alle Schlüssel des Projektes dargestellt. Im Normalfall wird daher ein Eintrag mit dem Schlüssel 'Startseite' dazu verwendet, einleitende Informationen über den Inhalt des Projektes zu geben, von denen aus dann über Querverweise weitere Texte erreicht werden können.

Querverweise

Sie haben nun eine Reihe von separaten Einträgen erzeugt, die alle über einen oder mehrere Schlüsselbegriffe angesprochen werden können. Momentan besteht aber noch kein Zusammenhang zwischen diesen Einträgen. Dies kann über das Einfügen von Querverweisen geändert werden; auf diese Weise bietet sich während des Lesens eines Eintrags die Möglichkeit, in einen anderen Text mit verwandten Informationen zu wechseln.

Um Texte mit Querverweisen zu versehen, gehen Sie vor wie beim Markieren der Schlüssel - Sie wählen lediglich statt 'Schlüssel' die Bereichsmarke 'Querverweis' aus. Gehen Sie dann die Einträge nacheinander durch. Wenn Sie irgendwo auf einen Begriff stoßen, den Sie anderswo als Schlüssel markiert haben, können Sie aus dem Begriff einen Querverweis machen. Wird der Verweis später bei der Anzeige des Textes ausgewählt, wird der zu dem entsprechenden Schlüssel gehörende Eintrag angezeigt. Genauso können Sie natürlich noch explizit Querverweise einfügen, in denen Sie auf Einträge mit verwandten Informationen hinweisen. Grundregel ist jedoch, daß zu jedem Querverweis auch ein Schlüssel gleichen Namens existieren muß!

Grafiken

Hilfstexte können durch das Einfügen von Grafiken wesentlich attraktiver gestaltet werden. Als erstes müssen Sie dazu die entsprechende Grafik zeichnen. Bedienen Sie sich dazu bitte eines handelsüblichen Grafikprogrammes. Speichern Sie dann die fertige Grafik ab. Am sinnvollsten ist es, wenn Sie dazu im Verzeichnis mit den Projekt-Quelltexten ein Unterverzeichnis anlegen, in dem dann alle Grafiken des jeweiligen Projektes abgespeichert werden können.

Wählen Sie dann aus dem 'Kennung einfügen'-Untermenü den Punkt 'Grafik'. Der HotHelpMarker öffnet ein Fenster, in dem Sie den Namen der Grafikdatei eingeben oder über ein Dateiauswahlfenster komfortabel auswählen können. Unter dem Eingabefeld wird eine Vorschau der ausgewählten Grafik angezeigt.

Schriftarten

Um bestimmte Teile des Textes noch besonders hervorzuheben (Überschriften, wichtige Passagen, ...), können Sie diese fett, kursiv, unterstrichen oder in jeder beliebigen Kombination dieser Attribute darstellen lassen. Dies wird über das Schriftarten-Untermenü im Bearbeiten-Menü gesteuert.

Möchten Sie z.B. alle Überschriften fett und unterstrichen ausgeben, gehen Sie dazu wie folgt vor: Sie schalten zuerst im Bereichsmarken-Untermenü die noch von vorhin gesetzte Marke aus, indem Sie sie nochmals anwählen. Die Anzeige der aktuellen Bereichsmarke in der Titelleiste des Fensters wird daraufhin gelöscht. Danach wählen Sie im Schriftarten-Menü nacheinander die Punkte 'Unterstrichen' und 'Fett' aus. In der Titelleiste erscheinen die Kürzel 'U' und 'F' für diese beiden Schriftarten. Nun setzen Sie den Cursor an den Anfang einer Überschrift und markieren Sie diese wie gehabt (die Bereichsmarke mußte vorher ausgeschaltet werden, da ansonsten die Überschrift nicht nur fett und unterstrichen, sondern auch noch mit dieser Bereichsmarke markiert worden wäre...).

Sie können auf diese Weise jedem Textbereich beliebige Textattribute zuordnen. Um ein solches Attribut wieder zu löschen, wählen Sie im Schriftarten-Menü 'Normaler Text' aus und markieren den entsprechenden Bereich. Wollen Sie die Textattribute beim Markieren unverändert lassen, schalten Sie einfach alle Optionen im Schriftarten-Menü wieder aus (die Anzeige in der Titelleiste ist dann leer).

Die Übersetzung

Damit haben Sie die grundlegenden Arbeitsschritte kennengelernt, um ein einfaches Projekt zu erstellen. Wählen Sie nun aus dem Projekt-Menü 'Speichern' aus, so sichert der HotHelp-Textmarker die Datei einschließlich aller Änderungen auf Diskette bzw. Festplatte. Falls Sie für Ihr Projekt mehrere Quelldateien erstellt haben, sollten Sie nun die übrigen Dateien in ähnlicher Weise behandeln. Sind alle Dateien mit Marken versehen und wieder abgespeichert, kann die Übersetzung beginnen.

Dieser Schritt ist recht einfach: wählen Sie aus dem Übersetzen-Menü im Untermenü 'Dateien übersetzen' den Punkt 'Alle ...'. Der HotHelp-Textmarker lädt nun den HotHelp-Übersetzer nach, der die eigentliche Übersetzung durchführt. Ist die Übersetzung abgeschlossen, können Sie das Fenster des Übersetzers wieder schließen. Falls Fehler aufgetreten sind, öffnet sich automatisch das Überset-

zungsfehler-Fenster, in dem alle Fehler aufgelistet sind. Sie können von hier aus die fehlerhaften Zeilen einfach editieren.

Haben Sie alle Fehler beseitigt, wählen Sie am besten den Menüpunkt 'Geänderte ...' aus dem 'Dateien übersetzen'-Untermenü. Im Gegensatz zu 'Alle ...' werden hierbei nur die Dateien übersetzt, die seit dem letzten Übersetzungsvorgang noch geändert worden sind. Diese Option ist recht praktisch, wenn Sie mit vielen Quelldateien arbeiten, da erfahrungsgemäß immer nur in einigen Dateien Änderungen vorgenommen werden. Der Übersetzer verläßt sich allerdings darauf, daß die Systemzeit immer korrekt gestellt ist - ist das bei Ihnen nicht der Fall, kann es zu unerwünschten Effekten kommen. Sie sollten dann besser auf diese Option verzichten!

10.5.5. Resümee

Natürlich läßt sich die Erstellung eines Projektes nicht immer genau so in Schritte einteilen, wie dies hier beschrieben wurde. Schritt 2 (Schreiben der Texte) und Schritt 3 (Markieren und Übersetzen) werden wahrscheinlich häufig wiederholt nacheinander ausgeführt, da es erfahrungsgemäß nicht möglich ist, alle gewünschten Texte auf einmal einzugeben. Oftmals stellt sich erst später heraus, daß einige Gebiete nur ungenügend behandelt oder ganz ausgelassen wurden, obwohl sie für das Thema des Projektes noch von Interesse sind. Beim Verteilen der Querverweise ergibt sich oft der Bedarf nach neuen Einträgen mit ergänzenden Informationen, die dann durch diese Querverweise angesprochen werden sollen.

Wenn sie einen Text wieder in den Editor laden, den Sie vorher im HotHelp-Textmarker mit Marken versehen haben, werden Sie feststellen, daß der Text geändert wurde - der HotHelp-Textmarker hat eine Reihe von Kennungen eingefügt, mit denen die Markierung der im Editorfenster sichtbaren Bereiche in der Datei festgehalten wird. Tatsächlich handelt es sich hierbei um die Marken, die unter Version 2.00 von HotHelp noch von Hand in den Text eingefügt werden mußten! Wenn Sie einen mit solchen Marken versehenen Text von einem Editor aus ändern, sollten Sie darauf achten, die Kennungen selber nicht zu verändern - andernfalls kann es beim nächsten Mal, wenn die Datei vom HotHelp-Textmarker geladen wird, zu Problemen kommen!

Wenn Sie die Arbeit an Ihrem Projekt abgeschlossen haben, sollten sie über den Menüpunkt 'Projekt überprüfen' aus dem Übersetzen-Menü einen Test Ihres Projektes durchführen. Dabei werden Sie z.B. darauf hingewiesen, wenn ein Begriff mehrfach als Schlüssel definiert wurde, wenn zu einem Querverweis kein passender Schlüssel existiert oder wenn gar mehrere passende Schlüssel existieren. Dies alles kann auf logische Fehler (oder auch einfach Schreibfehler) in Ihrem Projekt hindeuten.

10.5.6. Weitere Gestaltungsmöglichkeiten

Hier werden einige weitergehende Möglichkeiten erläutert, um Hilfstexte zu gliedern und in eine bestimmte Struktur zu bringen.

Titel

An jeden Eintrag kann ein Titel vergeben werden, der bei der Anzeige des Textes in der Titelzeile des HotHelp-Fensters hinter dem Projekt- und dem Schlüsselbegriff angezeigt wird.

Kapitel, Vorgänger und Nachfolger

Diese drei Bereiche wirken sich bei der Darstellung auf das Kapitel- und die beiden Pfeil-Aktionssymbole des HotHelp-Fensters aus. Sie werden im Prinzip genauso verwendet wie Querverweise, allerdings sind die so markierten Bereiche nicht mehr im übersetzten Text sichtbar. In jedem Eintrag dürfen diese drei Bereiche jeweils höchstens einmal auftreten.

Während normale Querverweise im Hilfstext sichtbar sind und über die Maus oder die Tastatur ausgewählt werden, benutzt HotHelp die über diese drei Marken definierten Querverweise, wenn das Kapitel- oder eines der beiden Pfeil-Aktionssymbole vom Anwender betätigt wird. Sie können also auf diese Weise jedem Text eine Kapitelübersicht, einen Vorgänger und einen Nachfolger zuordnen.

Projekt- und ARexx-Querverweise

Über Projekt-Querverweise kann das Problem mehrerer gleichlautender Schlüssel in verschiedenen Projekten gelöst werden. Möchten Sie z.B. einen Querverweis auf das Wort 'Task' einfügen, ergibt sich das Problem, daß dieser Schlüssel sowohl im Projekt 'Glossar' als auch in 'Libs 3.0' auftaucht - natürlich mit gänzlich unterschiedlichen Bedeutungen. In diesem Fall können Sie über den Projekt-Querverweis auch noch den Namen des Projektes angeben, auf das sich der Querverweis beziehen soll.

ARexx-Querverweise ermöglichen es, ARexx-Kommandos in einen HotHelp-Text einzubinden. Wählt der Anwender dann den Querverweis aus, wird der Befehl an ARexx übermittelt und ausgeführt. Wahlweise kann auch ein ARexx-Host angegeben werden, an den das ARexx-Kommando dann geschickt wird. Voraussetzung dafür ist allerdings, daß ARexx auf Ihrem System installiert ist!

Schlüsselgruppen

Schlüsselgruppen stellen eine Möglichkeit dar, die Entwicklung umfangreicher Projekte mit mehreren Gliederungsstufen (Kapitel, diesen zugeordnete Unterkapitel, etc.) zu erleichtern. Jedes Projekt kann über bis zu 255 unterschiedliche Schlüsselgruppen verfügen; jeder Schlüssel kann einer oder mehreren dieser Gruppen zugeteilt werden.

Solche Gruppen können nun auf dreierlei Weisen verwendet werden. Über die Gruppenlisten-Kennung kann eine Übersicht aller Schlüssel einer oder mehrerer Gruppen in den Text eingefügt werden. Jeder der Schlüssel wird als Querverweis angezeigt, so daß der zugehörige Text vom Benutzer auf die übliche Weise ausgewählt werden kann.

Weiterhin kann jeder Gruppe ein Titel (ein sog. Gruppentitel) zugeordnet werden. Wird ein Hilfstext zu einem Schlüssel angezeigt, der sich in einer mit einem Titel versehenen Schlüsselgruppe befindet, wird dieser Gruppentitel in der Titelleiste des Fensters zwischen dem Projekt- und dem Schlüssel-Namen angezeigt. Betätigt der Benutzer das Kapitel-Aktionssymbol im HotHelp-Fenster, wird der Gruppentitel als Schlüsselbegriff verwendet und der entsprechende Hilfstext angezeigt, sofern ein Schlüssel dieses Namens in dem Projekt definiert wurde. Andernfalls wird die Startseite

des Projektes angezeigt. Die Wirkung dieses Gruppentitels kann durch explizite Angabe einer Kapitel-Marke (s.o.) überschrieben werden: eine mit der Kapitel-Marke definierte Verknüpfung hat Vorrang vor der durch einen Gruppentitel hergestellte Beziehung.

Gruppentitel können über den entsprechenden Menüpunkt im Projekt-Menü editiert werden. Bitte beachten Sie, daß zwischen den normalen Titeln und den Gruppentiteln keinerlei Zusammenhang besteht! Normale Titel werden lediglich in der Titelleiste des Fensters angezeigt, haben jedoch keinerlei weitere Wirkung!

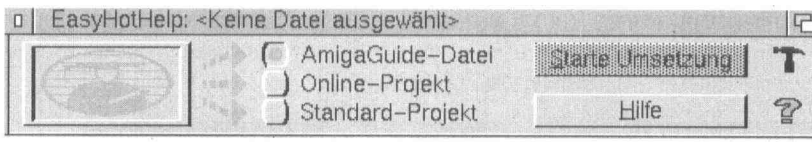
Gruppen können außerdem dazu dienen, das Problem mehrerer gleichlautender Schlüssel zu lösen. Normalerweise sollte ein Projekt zwar nicht mehrere Schlüssel mit gleichem Namen, aber unterschiedlichen Texten beinhalten. Ist dies aber doch einmal der Fall (wie es sich z.B. bei dem Projekt Libs & Devs nicht umgehen ließ, wo es Überschneidungen zwischen Struktur-, Funktions- und Konstantennamen gibt), stellt sich die Frage: Wie kann ein Querverweis sich eindeutig auf einen der gleichlautenden Schlüssel beziehen?

Die Antwort: Jeder der Querverweis-Marken kann ebenfalls eine Gruppennummer zugeordnet werden. Geben Sie nun demjenigen der gleichlautenden Schlüssel, den Sie durch einen Querverweis ansprechen möchten, eine eindeutige Nummer, können Sie sich im entsprechenden Querverweis über diese Nummer direkt auf den gewünschten Schlüssel beziehen.

Schlüsselgruppen sind keine sehr einfache Materie und da ein Beispiel mehr sagt als tausend Worte, soll hier nur noch auf das zweite der beiden Beispielprojekte verwiesen werden. Es zeigt an dem leicht nachvollziehbaren Thema von Kochrezepten mit vielen erklärenden Texten, wie Gruppen sinnvoll in HotHelp-Projekten angewendet werden können. Am besten starten Sie jetzt den HotHelpMarker, machen das Verzeichnis 'BeispielProjekt2' zum Projektverzeichnis und arbeiten die Dateien in diesem Verzeichnis der Reihe nach durch (beginnen Sie mit der Startseite und fahren dann mit der italienischen Küche fort).

10.6. EasyHotHelp

EasyHotHelp ist ein Werkzeug, mit dessen Hilfe Sie sehr einfach schon bestehende Dokumente als HotHelp-Projekte anmelden können. Sie können damit alle Textdateien, die im ASCII-Format vorliegen (speziell auch AmigaGuide-Dateien), sehr einfach umsetzen - z.B. die Anleitungen von Public-Domain-Programmen. Voraussetzung für die Verwendung dieses Programms ist allerdings, daß Sie mindestens Betriebssystemversion 2.0 verwenden.



Nach dem Start öffnet das Programm auf der Workbench ein kleines Fenster. Ergreifen Sie nun das Piktogramm des Textes, den Sie bearbeiten möchten, und legen es in das EasyHotHelp-Fenster. Das Piktogramm wird in dem Fenster dargestellt, und der Name der Datei erscheint in der Titelleiste. Sie können diese Datei nun auf drei verschiedenen Wegen umsetzen, indem Sie den entsprechenden Druckknopf anwählen und dann die Umsetzung über das entsprechende Symbol starten:

- Als AmigaGuide-Datei. Dies empfiehlt sich natürlich nur, wenn es sich auch wirklich um eine Datei im AmigaGuide-Format handelt - ansonsten zeigt der Übersetzer Ihnen eine entsprechende Fehlermeldung an. AmigaGuide-Dateien haben üblicherweise die Endung '.guide'.
- Als Online-Projekt. Die Textdatei wird als Eintrag eines Online-Projektes verwendet (siehe auch Kapitel 7). Das Programm verhält sich dabei genauso, als ob Sie die Datei in einen der unterstützten Editoren geladen, als Block markiert und dann über <Ctrl>-<Amiga>-<s> übernommen hätten. Sie haben somit also die Möglichkeit, Online-Projekte zu definieren, auch wenn Sie über keinen der genannten Editoren verfügen.
- Als Standardprojekt. Wie schon in Kapitel 10.5 erläutert wurde, müssen - um aus einer normalen Textdatei ein Standardprojekt zu erzeugen - bestimmte Bereiche des Textes als Schlüssel, Querverweise, etc. markiert werden.

Um solche Markierungen in den Text einzusetzen, kann normalerweise der HotHelpMarker benutzt werden. Dieser unterstützt zwar alle Möglichkeiten der Projekterzeugung, kann durch seine Vielfalt an Funktionen für den Einsteiger jedoch auch sehr verwirrend sein. Sie finden daher hier eine Art 'kalorienarmer HotHelpMarker light'.

EasyHotHelp stellt den geladenen Text dazu in einem weiteren Fenster dar, wo Sie durch einfaches Anklicken und Ziehen mit der Maus Schlüssel, Querverweise und Eintragsende setzen und den fertig bearbeiteten Text dann sofort übersetzen können. Im Falle von Übersetzungsfehlern können Sie diese anzeigen lassen und korrigieren.

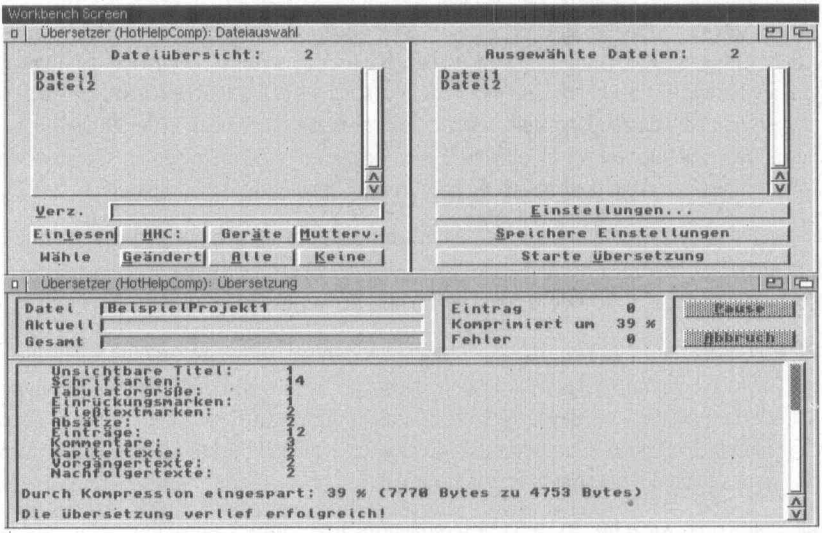
Sie können den so markierten Text auch später noch mit dem HotHelpMarker weiterbearbeiten - EasyHotHelp erzeugt ein temporäres Verzeichnis (üblicherweise in der RAM-Disk, Sie können

jedoch auch ein anderes Verzeichnis wählen), in dem der Text mit seinen Markierungen im üblichen Format abgelegt wird.

Weitere Informationen über EasyHotHelp können Sie wie üblich nach dem Start des Programms über die Hilfetaste bzw. die Hilfe-Symbole und -Menüs abfragen.

10.7. HotHelpComp - der HotHelp-Übersetzer

Dieses Programm entspricht dem 'Projekte übersetzen'-Modul des alten HotHelpManagers der Version 2.00 von HotHelp. Es wird bei der Erstellung eigener Projekt verwendet, um die mit Textmarken versehenen Quelltexte in HotHelp-Projekte zu übersetzen.



Das Programm ist ebenfalls in mehrere Fenster gegliedert, die zum einen dem eigentlichen Übersetzungsvorgang dienen, zum anderen jedoch auch weitreichende Möglichkeiten der freien Gestaltung der Markierungen bieten, die in den Quelltexten besondere Bereiche (Querverweise, Schlüssel, Schriftarten, Absatzenden etc.) kennzeichnen.

Da HotHelpComp üblicherweise im Rahmen der Übersetzung vom HotHelpMarker aufgerufen und kontrolliert wird, soll an dieser Stelle nicht weiter auf die Bedienung dieses Programms eingegangen werden.

Der "Power-User", dem das Herumwirbeln mit sinistren HotHelp-Steuercodes in Dutzenden von Quelldateien in MegaByte-großen Projekten nicht über den Kopf wächst, und der seine HotHelp-Quelldateien lieber auf herkömmliche Weise mit dem Editor erzeugt und ändert, hat jedoch die Möglichkeit, den HotHelp-Übersetzer direkt aufzurufen, ohne den "Umweg" über den HotHelp-Textmarker gehen zu müssen (Ähnlichkeiten mit dem Autor des Libs&Devs-Projektes sind weder zufällig noch unbeabsichtigt).

11. Letzte Worte

Viele Dinge wurden in diesem Handbuch nur oberflächlich angerissen - und das aus gutem Grund: HotHelp soll Ihnen schließlich das endlose Blättern in Handbüchern ersparen - warum sollten wir Sie also mit einem dicken HotHelp-Handbuch belasten? Die Projekte 'HotHelp' und 'HotHelpTools' enthalten alle Informationen, die für die Arbeit mit HotHelp und seinen Zusatzprogrammen notwendig sind. Auf diese Weise steht Ihnen auf Tastendruck hin genau die Information zur Verfügung, die Sie gerade brauchen. Zögern Sie also nicht, diese Informationsmöglichkeit zu nutzen, ehe Sie der Funktion eines rätselhaften Bedienungselementes durch Probieren auf den Grund zu gehen versuchen!

Das HotHelp-System selbst wurde unter ständiger Nutzung von HotHelp programmiert (insbesondere das Projekt Libs & Devs wurde häufig zu Rate gezogen). Es ist also praxiserprobt, und wir selbst möchten bei der Arbeit nicht mehr darauf verzichten. Wir hoffen daher, daß HotHelp auch Ihnen Ihre Arbeit am Amiga erleichtert und Ihnen - gemäß dem Motto von HotHelp - wirklich schnelle Hilfe zu jeder Zeit bietet.

In diesem Sinne wünschen wir Ihnen viel Erfolg.

Die Autoren

Maxon C++ 3

Features

Light-Version
- C++-Compiler (AT&T 2.0 Standard)
- ANSI C-Compiler (Aztec- und Lattice-ANSI-kompatibel)
- CLI-Compiler
- HotHelp 3-Reader zum Anzeigen von HotHelp-Projekten

Developer-Version
- C++-Compiler (AT&T 3.0 Standard)
- Templates, Exceptions
- ANSI C-Compiler (wie Light-Version)
- CLI-Compiler
- beinhaltet zusätzlich: MaxonHotHelp 3, MaxonSDB, MaxonEASY-OBJECTS und MaxonASM-CLI

Compiler (C/C++)
- komfortable Projektverwaltung (Make)
- direkter Fehlerstellenansprung
- voll in den Editor integriert
- über ARexx-Schnittstelle steuerbar
- sehr schnelle Turnaroundzeiten
- echte vorkompilierte Include-Dateien (Preincludes)
- Include-Dateien für Kickstart 3.1
- speicherbarer Programmstatus
- Codegenerator für prozessorspezifischen Code 68010/20/30 und 68881/2 FPU

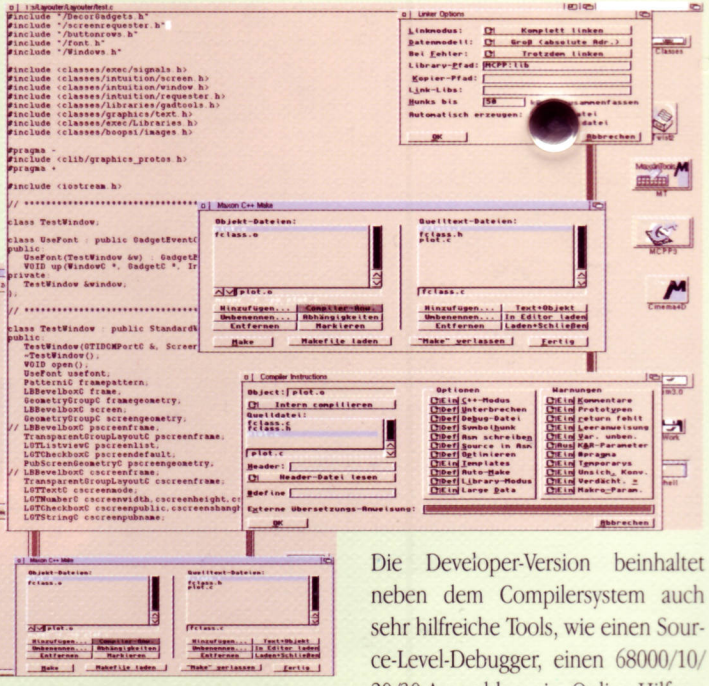
Programmeditor (EDWARD)
- Multi-Window-Technik
- beliebig viele Texte gleichzeitig bearbeitbar (speicherabhängig)
- Tastatur- und Menüfunktionen frei belegbar
- speicherbarer Programmstatus
- ARexx-Schnittstelle

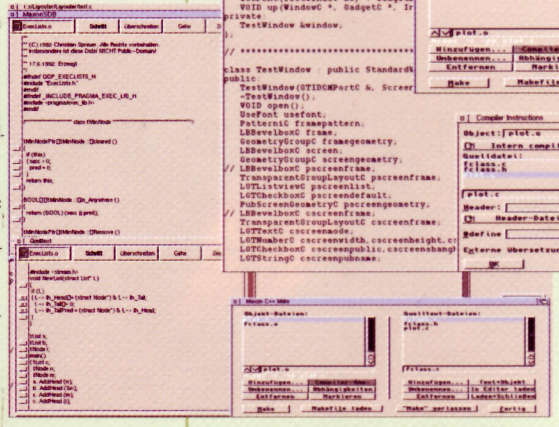
Hilfesystem HotHelp 3
(nur in der Developer-Version)
- Deutschsprachige Beschreibung der Library- und Device-Funktionen zu Amiga-OS 2.1
- jederzeit aktivierbar
- kontextsensitiv
- komfortabler Editor für eigene Projekte
- Online-Projekte (direkte übernahme von Texten und Sourcecode in das Hilfesystem)
- ARexx-Schnittstelle
- Grafikeinbindung direkt im Hilfetext
- direktes Lesen von AmigaGuide™-Projekten
- lizenzfreie Weitergabe des HotHelp 3-Anzeigers

Features der Klassenbibliothek (MaxonEASY-OBJECTS)
(nur in der Developer-Version)
- Ressourcenhandling: Betriebssystemressourcen (Speicher, Signale, Fenster, Bildschirme etc) werden automatisch verwaltet und bei Programmende freigegeben (keine aufwendige Fehlerbehandlung mehr nötig).
- Fehlerbehandlung durch Exceptions
- Handlerkonzept
- Datenstrukturen: Listen, dynamisch große Arrays, Datenpuffer (z.B. für I/O-Operationen)
- Verwaltung folgender Betiebssystemresourcen: Exec: Signale, Nachrichtenports, Speicher, Libraries, Devices, Listen - DOS: Locks, Dateien, Pfadnamen, Kommandozeilenargumente, Notifikation - Utility: Hooks und Taglisten - Intuition: Bildschirme, Fenster, IDCMP-Nachrichten, Systemrequester - Libraries: GadTools, ASL, Icons, Workbench, Tooltypes, Applikationenfenster
- BOOPSI Unterstützung
- automatisch größenveränderbare und font-veränderbare Fenster durch den Layouter
- deutschsprachige Online-Dokumentation

Externer Assembler (MaxonASM-CLI)
(nur in der Developer-Version)
- 68000/10/20/30 Makro-Assembler
- FPU 68881/2 und MMU-Unterstützung
- Multi-Pass-Optimierung erzeugt sehr schnellen und kompakten Programmcode.
- voll in die Projektverwaltung integrierbar

C/C++ Source Level Debugger (MaxonSDB)
(nur in der Developer-Version)
- Multi-Window-Technik
- einfache Bedienung - Drag&Drop
- Beobachten von Variableninhalten
- Haltepunkte
- direkt vom Compiler aus aufrufbar





MaxonC++ ist die erste konsequente Implementierung des C++ 3 Standards auf dem Amiga.

Das MaxonC++-Compilersystem beinhaltet sowohl einen ANSI C-, als auch einen zum AT&T 3.0-Standard kompatiblen C++-Compiler. Der Compiler erzeugt aus den C++-Quelltexten direkt ausführbare Programme ohne Umweg über C, Assembler oder andere Sprachen. Durch die volle Systemintegration werden überzeugend schnelle Turnaroundzeiten erreicht.

Die Developer-Version beinhaltet neben dem Compilersystem auch sehr hilfreiche Tools, wie einen Source-Level-Debugger, einen 68000/10/20/30-Assembler, ein Online-Hilfesystem mit der deutschsprachigen Beschreibung der Systemfunktionen und eine umfangreiche Klassenbibliothek.

Das auch in der Light-Version aufwendige und gut strukturierte deutschsprachige Handbuch ist ein kompletter ANSI-C/C++-Programmierlehrgang und auch für Einsteiger zu empfehlen.

Lauffähig auf allen Amigas mit min. 2 MB Speicher und Amiga-OS 2.0. Bei Nutzung der Klassenbibliothek werden 6 MB Speicher benötigt.

AMIGA®

4 010509 411302

MAXON Computer GmbH • Industriestr. 26
D-65760 Eschborn • Telefon 06196/481811
Irrtümer und Änderungen vorbehalten!

MAXON computer